MICROWAVE FILTER COMPUTER CODE

MUHAMMAD TAUFIQ BIN ABDUL RASHID

ELECTRICAL & ELECTRONICS ENGINEERING
UNIVERSITI TEKNOLOGI PETRONAS
JUNE 2010

MICROWAVE FILTER COMPUTER CODE

By

MUHAMMAD TAHERI BIN ABDUL RASHID

FINAL PROJECT REPORT

Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

MICROWAVE FILTER COMPUTER CODE

by

Muhammad Taheri bin Abdul Rashid

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:

_____

Prof. Grant Andrew Ellis

Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

June 2010

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

Muhammad Taheri bin Abdul Rashid

ABSTRACT

A microwave filter is a two port network used to control the frequency response in a microwave system by providing transmission at frequencies within the pass-band of the filter and attenuation in the stop-band of the filter. Typical frequency response include low-pass, high-pass, band pass, and band reject characteristics. The filters will be designed using the insertion loss method. It uses network synthesis techniques to design filters with a completely specified frequency response. The design is simplified by beginning with low-pass filter prototypes that are normalized in terms of impedance and frequency. Transformations are then applied to convert the prototype designs to the desired frequency range and impedance level. The values needed for the lumped element warrant further research in code to be developed for faster analysis. The lumped element values can then be realized by using Richard's Transformation and Kuroda's Identity in order to design a working filter.

ACKNOWLEDGEMENTS

First of all, I'm grateful to Allah SWT blessings for giving me a chance and strength to complete my Final Project Report for June 2010. I have learned a lot in terms of programming, which somehow is the first experience for me, because I'm always living a student life. I gain a lot of valuable experience in Python language.

Many thanks also go to Prof. Grant Andrew Ellis, Project Supervisor for giving me guidance and helping me to gain information and knowledge about the Python Language for developing the microwave filter computer code.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

CHAPTER 1
INTRODUCTION

1.1 Background of Study

This section briefly explains filter design by the insertion loss method. Beforehand, an explanation about the filter will be discussed. Moreover, the emphasis is more on the filter design by the insertion loss method.

In designing a filter, there are 2 popular methods, which is filter design by image parameter method and filter design by the insertion loss method. For the scope of my project, it will be mainly about the filter design by the insertion loss method.

Using this method, there is some process involved before the filter can actually be designed for use in practical work. The process includes following:

1. Determining the frequency, ω (Hz) required for the filter
2. Determining the cutoff frequency, ω_c (Hz) required for the filter
3. Determine the attenuation (dB) desired for the given cutoff frequency.
4. Determine the filter order, n given information from 1,2, and 3
5. Specify the output resistance, Z_o (ohm)
6. Specify types of filter whether low pass, high pass, band pass, or band stop filter
7. Specify types of filter response to use whether maximally flat response, maximally flat time delay response, or equal ripple response.

8. Determine prototype values given information from 4 and 7.
9. Call in function that is suitable given all the criteria above.

In order to design the filter efficiently, the values entered will produce different kinds of the L and C networks. Commonly the values of Z_o are small enough, for example 50 ohm. Moreover, by specifying the output impedance, we know exactly what we are working for.

For the purpose of this project, only filter design by the insertion loss method will be focused on. This is due to the fact that we can work in many ways so we can get the results in terms of not only low pass filter, but also including high pass, band pass, and band stop filter.

During the process of designing, the L and C value we get from the low pass filter can be transformed into the high pass, band pass, and band stop filter simply by using the impedance and frequency scaling, followed by band pass or band stop transformation where it is applicable. Prototype filter transformations are basically the modified value of low pass counterparts, with specific transformations pre-determined.

A Python program will be written and coded in order to simplify tedious work of normal calculation, where it is prone to error and confusion. It can function as a stand alone or as the guide for engineer while calculating themselves and compared with the actual values. With that being said, it is basically built with no error in terms of mathematic calculation as it is not limited in terms of its function, which includes ideal response.

Hence, it is more preferred and desired instead of normal calculations where it can take a long time and still prone to error. At the end of the process, it will generate

the values and results for each of the lumped elements involved in making the filter or in another words, designing the filter itself.

This lumped element values typically cannot be realized into the desired capacitor and transistor. This is due to the manufacturer limitations. Therefore, these values will be transformed to the micro strip equivalent of its values and is suitable to use in real life application where there is a need to use a large number of filter banks.

1.2 Problem Statement

In the process for designing a filter, the design is prone to error and can also take significant amount of time to compute. This can lead to the poor filter response with inaccurate value of L and C. This will affect the filter thoroughly when the filter is filtering the incoming signal. There will be significance amount of unwanted signals at the end of the filter which is undesired. The main problem is the consistency of the L and C which will determine the overall response of the filter. This is easily overcome by the help of a program to assist in designing the filter. The lumped element values can then be realized by using Richard's Transformation and Kuroda's Identity in order to design a working filter.

1.3 Objectives and Scope of Study

The main objective of this project is to develop an easy-to-use and understandable program that will assist the users in the filter design by the insertion loss method. It will also provide the user friendly interface, with the working code that will be explained thoroughly at the source level. For this design purpose, it will incorporate the following statistical method:

1. Determine the filter order.
2. Determine prototype values.
3. Call in function that is suitable given all the criteria.
4. Calculation of the value of *L1, C2, L3, C4* ... (Type I Filter)
5. Calculation of the value of *C1, L2, C3, L4* ... (Type II Filter)
6. The value for high pass, band pass, and band stop filter using the impedance and frequency scaling, followed by band pass and band stop transformations respectively.
7. Richard's Transformation and Kuroda's Identity for filter realization.

CHAPTER 2
LITERATURE REVIEWS

2.1 Filters Design Using the Insertion Loss Method

The objective of this project is to develop a microwave filter computer code. A microwave filter is a two port network use to control the frequency response a certain point in a microwave system by providing transmission at frequencies within the pass-band of the filter and attenuation in the stop-band of the filter. Typical frequency response include low-pass, high-pass, bandpass, and band reject characteristics.

For this project, the code revolves around the filters designed using the insertion loss method. It uses network synthesis techniques to design filters with a completely specified frequency response. The design is simplified by beginning with low-pass filter prototypes that are normalized in terms of impedance and frequency.

Transformations are then applied to convert the prototype designs to the desired frequency range and impedance level. The insertion loss method provides lumped-element circuit. This design must be modified to use distributed elements consisting of transmission line section for microwave applications. The Richard's transformation and Kuroda identities provide this step.

The perfect filter would have zero insertion loss in the passband, infinite attenuation in the stopband, and a linear phase response in the pass-band to avoid signal distortion. Such filters do not exist in practice, so compromises must be made, where here lays the art of filter design. The insertion loss method allows a high

degree of control over the passband and stopband amplitude and phase characteristics, with a systematic way to synthesize a desired response.

The necessary design trade-offs can be evaluated to best meet the application requirements. If, for example, a minimum insertion loss is most important, a binomial response could be used, a Chebyshev response would satisfy a requirement for the sharpest cut-off.

If it is possible to sacrifice the attenuation rate, a better phase response can be obtained by using a linear phase filter design. And in all cases, the insertion loss method allows filter performance to be improved in a straightforward manner, at the expenses of a higher order filter.

In the insertion loss method a filter response is defined by its insertion loss, or power loss ratio, P_{LR}:

$$P_{LR} = \frac{\text{Power from source}}{\text{Power to load}} = \frac{P_{inc}}{P_{load}} = \frac{1}{1-|\Gamma(\omega)|^2}.$$

The insertion loss (IL) in dB is

$$IL = 10 \log P_{LR}$$

$|\Gamma(\omega)|^2$ is an even function of ω, it can be expressed as a polynomial in ω^2.

$$|\Gamma(\omega)|^2 = \frac{M(\omega^2)}{M(\omega^2) + N(\omega^2)}.$$

where M and N are real polynomials in ω^2.

$$P_{LR} = 1 + \frac{M(\omega^2)}{N(\omega^2)}$$

For a filter to be physically realizable, its power loss ration must be of the form above. Specifying the power loss ratio simultaneously constrains the reflection coefficient, $\Gamma(\omega)$. There are some practical filter response which is maximally flat, equal ripple, and linear phase.

For maximally flat, this characteristic also called the binomial or Butterworth response, and is optimum in the sense that it provides the flattest possible passband response for a given filter complexity, or order (N). For a low pass filter, it is specified by

$$P_{LR} = 1 + k^2 (\omega/ \omega_c)^{2N}$$

Where N is the order of the filter and ω_c is the cutoff frequency. The passband extends from $\omega=0$ to $\omega= \omega_c$, at the band edge the power loss ratio is $1 + k^2$. This is the 0dB point, as is common, k=1. For $\omega> \omega_c$,the attenuation increase monotonically with frequency. For $\omega>> \omega_c$, $P_{LR} = k^2 (\omega/ \omega_c)^{2N}$, which shows that the insertion loss increase at the rate of 20N dB/decade. Like the binomial response for multi section quarter-wave matching transformers, the first (2N-1) are zero at $\omega=0$.

For equal ripple, if a Chebyshev polynomial is used to specify the insertion loss of an N-order low pass filter as

$$P_{LR} = 1+k^2T_N^2(\omega/ \omega_c)$$

Then a sharper cut-off will result, although the passband response will have ripples of amplitude $1+ k^2$, since $T_N(x)$ oscillates between ±1 for $|x| \leq 1$. Thus k^2

determines the passband ripple level. For large x, $T_N(x) = \frac{1}{2}(2x)^N$, so for $\omega >> \omega_c$ the insertion loss becomes

$$P_{LR} = k^2/4\ (2\omega/\ \omega_c)^{2N}$$

Which also increase at the rate of 20N dB/decade. But the insertion loss for the Chebyshev case is $(2^{2N})/4$ greater than the binomial response, at any given frequency where $\omega >> \omega_c$[1].

It is important to have a linear phase response in the passband to avoid signal distortion for some applications such as multiplexing filters for communication systems. Since a sharp cut-off response is generally incompatible with a good phase response, the phase response of a filter must be deliberately synthesized, usually resulting in inferior attenuation characteristics. More general filter specifications can be obtained, but the above cases are the most common. The design of low pass filter prototypes is normalized in terms of impedance and frequency.

This normalization simplifies the design of filters for arbitrary frequency, impedance, and type low-pass, high-pass, bandpass, and band stop. The low-pass prototypes are then scaled to the desired frequency and impedance, and the lumped-element components replaced with distributed circuit elements for implementation at microwave frequencies.

To successfully design a microwave filter, all of the equations must be coded into computer language programming. This will be done using Python language. A thorough search will be made through the Internet and from existing libraries to collect all available information on the use of the Python programming to design a microwave filter.

The collection of modules needs to be taken into consideration when developing a working code. Analysis and testing of the code will be run through out to make sure it is flawless and working correctly. The results of the analysis will be used to design the microwave filter as specified by the user.

A useful manual will be written to make it user friendly so that it is explainable and it is in a presentable manner. This application is very useful in designing the microwave filter which all the specifications can be entered manually by the user.

2.2 Python Programming Language

Python is a remarkably powerful dynamic programming language that is used in a wide variety of application domains. Python is often compared to Tcl, Perl, Ruby, Scheme or Java. Some of its key distinguishing features include:

- very clear, readable syntax
- strong introspection capabilities
- intuitive object orientation
- natural expression of procedural code
- full modularity, supporting hierarchical packages
- exception-based error handling
- very high level dynamic data types
- extensive standard libraries and third party modules for virtually every task
- extensions and modules easily written in C, C++ (or Java for Jython, or .NET languages for IronPython)
- embeddable within applications as a scripting interface

Fans of Python use the phrase "batteries included" to describe the standard library, which covers everything from asynchronous processing to zip files. The language itself is a flexible powerhouse that can handle practically any problem domain.

Python allows you write the code you need, quickly. And, thanks to a highly optimized byte compiler and support libraries, Python code runs more than fast enough for most applications.

The Python implementation is under an open source license that makes it freely usable and distributable, even for commercial use. The Python license is administered by the Python Software Foundation.

Basically Python software which is scripting language will be used to write the code throughout this project. It will be written into several modules which all have their basic functionality and when combined together, provide the functional program.

One of the main advantages of the Python is that the source code is freely available on the Internet from Python Software Foundation. As the result, neither royalties nor licenses need to be paid for distribution of any application by this language, library package upgraded and bug-fixed.

Furthermore, for the development of GUI, Python has been made more easily compared to the good old platform for programming. Making the code with the user interface is easier and quicker. Graphical interface help the programmers to compile and test run their program before the actual code finalized. Hence, error checking in terms of coding will also be carried out automatically [2].

CHAPTER 3
METHODOLOGY

3.1 Procedure Identification

The project work for the software development is show in Figure 1 below.

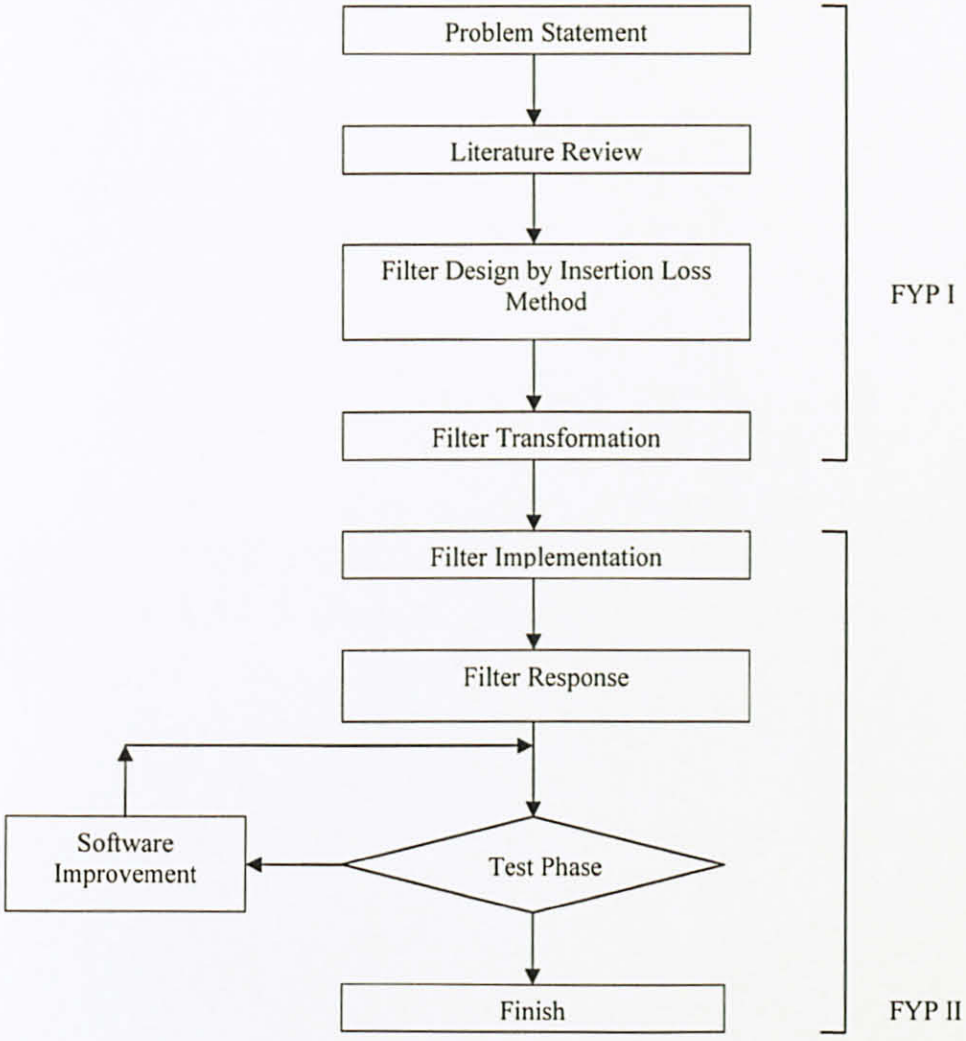

Figure 1: Project Flow Chart

3.2 Determining Filter Types

The project is organized by first determining the filter type whether it is low pass filter, high pass filter, bandpass filter, or a band stop filter. After that, the filter response is analyzed which is its response whether it is Butterworth or Chebyshev response, and the frequency and impedance is noted down.

With all the information gathered, the program will look up table for low pass filter prototype. Next, impedance scaling will be done. Hence, filter type transformation is performed to convert the filter into its desired response which is low pass filter, high pass filter, bandpass filter, or a band stop filter. Lastly, it will produce the component values for the lumped element accordingly. This is used to construct the filter desired. This is as shown in Figure 2.

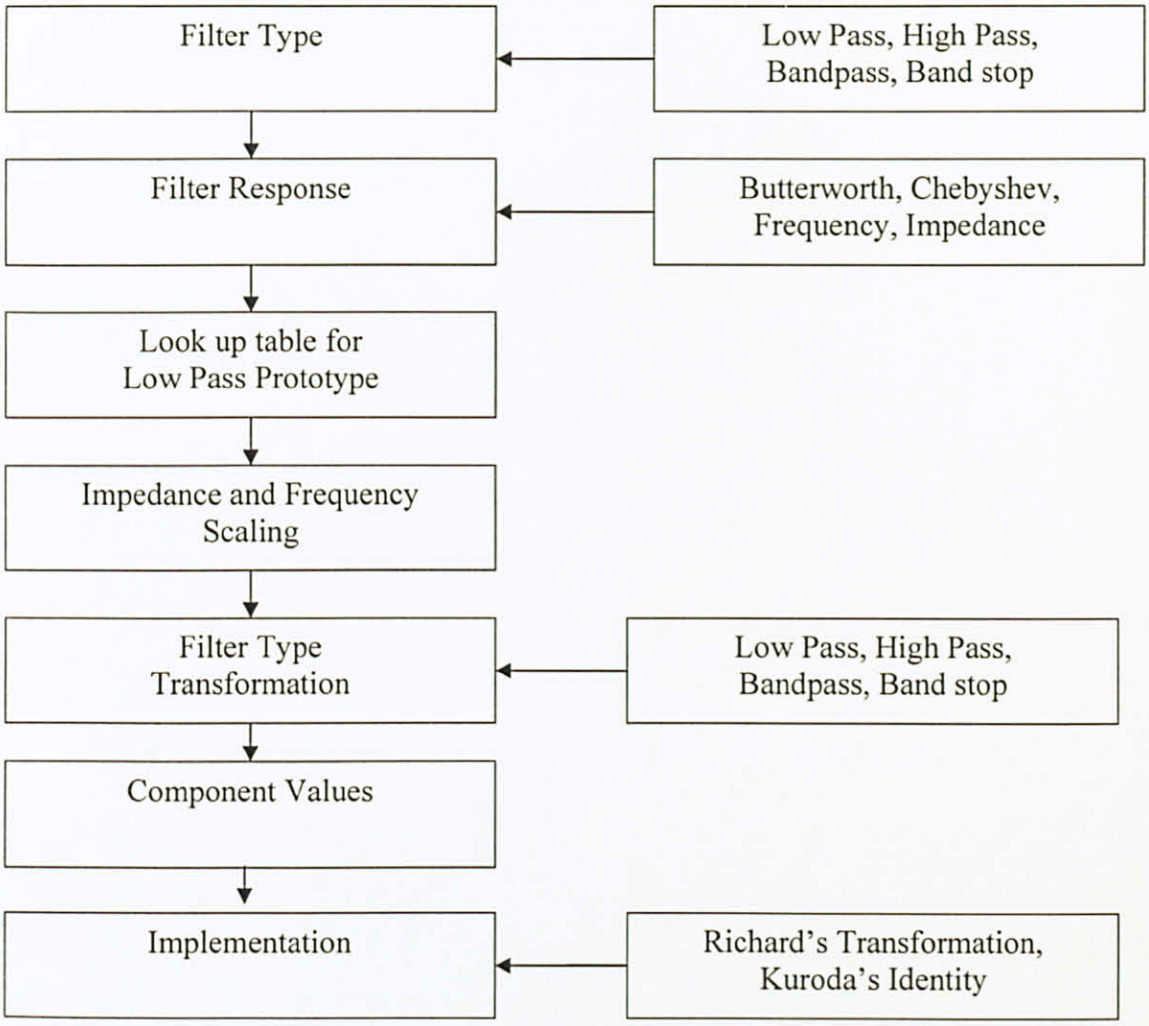

Figure 2: Determining Filter Types Step

3.3 Low Pass Filter Prototype

The first step is to design the low pass filter prototype. The low pass filter prototype is resembled by the following figure:

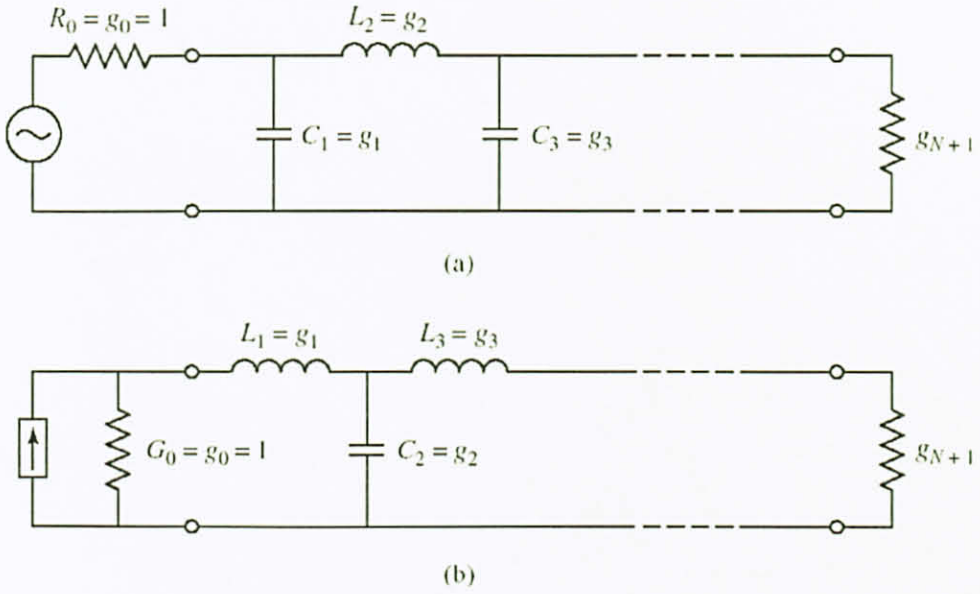

Figure 3: Low Pass Filter Prototype

As shown in the figure above, the inductor, L and the capacitor, C values will be given directly from the output. However the value for inductors and capacitors are only true for the low pass filter that has the impedance of 1 ohm and radian frequency, ω_c of 1 rad/sec. This is what we called low pass filter prototype.

The figure above shows two types of filter that can be used which is Type I Filter and Type II Filter. The only difference between the two is the alternating capacitor and inductor used. The purpose of having the two designs is for having the user to choose the closer value that is realistic to be used.

3.4 Impedance and Frequency Scaling

From here, we can do impedance scaling to match the impedance value for other value determined by the user. After that, we can do frequency scaling to transform from 1 rad/sec to another frequency specified by the user. After that, we can do filter type transformation to transform from low pass filter to either high pass filter, bandpass filter, or a band stop filter.

For the impedance and frequency scaling, it will be give as the following formula:

$$L' = (R_o L)/\omega_c$$
$$C' = C/(R_o \omega_c)$$
$$R_s' = R_o$$
$$R_L' = R_o R_L \quad (1)$$

where L, C, and R_L are the component values for the original prototype.

$$\Delta = (\omega_2 - \omega_1)/\omega_o$$
$$\omega_o = \sqrt{(\omega_2\ \omega_1)} \quad (2)$$

3.5 Filter Type Transformation

From here, the low pass filter can be transformed to high pass filter, bandpass filter, or a band stop filter. The transformation is basically changing the lumped elements represented by low pass filter to another lumped elements corresponding to each type of another filter. This can be realized from the table below:

Table 1: Summary of Prototype Filter Transformations

Low Pass	High Pass	Band Pass	Band Stop
L	$1/(\omega_c L)$	Series element of $L/(\omega_o \Delta)$ and $\Delta/(\omega_o L)$	Shunt element of $(L\Delta)/\omega_o$ and $1/(\omega_o L\Delta)$
C	$1/(\omega_c C)$	Shunt element of $\Delta/(\omega_o C)$ and $C/(\omega_o \Delta)$	Series element of $1/(\omega_o C\Delta)$ and $(C\Delta)/\omega_o$

3.6 Tools and Equipment Required

A computer with Python's Integrated Development Environment software installed. Python module MatPotLib and Numpy also is installed.

CHAPTER 4
RESULTS AND DISCUSSION

4.1 Element Values Table

Basically the code is broken into following parts, which is following:

1. Determining the frequency, ω (Hz) required for the filter
2. Determining the cutoff frequency, ω_c (Hz) required for the filter
3. Determine the attenuation (dB) desired for the given cutoff frequency.
4. Determine the filter order, n given information from 1,2, and 3
5. Specify the output resistance, Z_o (ohm)
6. Specify types of filter whether low pass, high pass, band pass, or band stop filter
7. Specify types of filter response to use whether maximally flat response, maximally flat time delay response, or equal ripple response.
8. Determine prototype values given information from 4 and 7.
9. Call in function that is suitable given all the criteria above.

The code will do the following task:

1. Determine the filter order.
2. Determine prototype values.
3. Call in function that is suitable given all the criteria.
4. Calculation of the value of *L1, C2, L3, C4* ... (Type I Filter)
5. Calculation of the value of *C1, L2, C3, L4* ... (Type II Filter)

6. The value for high pass, band pass, and band stop filter using the impedance and frequency scaling, followed by band pass and band stop transformations respectively.
7. Richard's Transformation and Kuroda's Identity for filter realization.

The task to read in the element values table consists of 3 separate tables. The table that will be used is as follows:

1. Element Values for Maximally Flat Low-Pass Filter Prototypes
2. Element Values for Equal-Ripple Low-Pass Filter Prototypes (0.5dB ripple)
3. Element Values for Maximally Flat Time Delay Low-Pass Filter Prototypes

4.2 Python Code Description

A basic understanding will be needed to develop the source code. The algorithm used here is mainly the integer input from the user and the table transformed into else if function in Python. It is coded nicely. [3]

The first code as shown in Figure 4 is the main function definition which will take the input from the user for frequency, cut-off frequency, attenuation, calculate the normalized frequency, output resistance, filter type, and filter response.

```
######################################################################
#
# Main function definition
#
#
#
#
######################################################################

from math import sqrt

frequency_f =input("Enter the Frequency desired(Hz): ")
cutoff_frequency_f =input("Enter the Cutoff Frequency desired(Hz): ")
attenuation = input("Enter the Attenuation(dB) desired: ")

pi=3.141592654

frequency = 2*pi*frequency_f
cutoff_frequency = 2*pi*cutoff_frequency_f

normalized_frequency= abs(frequency / cutoff_frequency) - 1

output_resistance =input("Enter the Output Resistance(ohm) desired: ")
filter_type =input("Enter which filter type [1],[2],[3],or [4] is desired:\n\n[1
filter_response =input("\nEnter which filter response [1],[2],or [3] is desired:
```

Figure 4: Main Function Definition

The next code as shown in Figure 5 will determine the filter order based on the user specification of frequency, cut-off frequency, and attenuation desired.

```
#####################################################################
#
# Code below will determine the filter order
# based on the frequency, cutoff frequency, and attenuation entered
#
#
#
#####################################################################
if pi == 3.141592654:

      if normalized_frequency > 1:
            if attenuation < 7: filter_order = 1
            elif attenuation < 12: filter_order = 2
            elif attenuation < 18: filter_order = 3
            elif attenuation < 24: filter_order = 4
            elif attenuation < 30: filter_order = 5
            elif attenuation < 37: filter_order = 6
            elif attenuation < 42: filter_order = 7
            elif attenuation < 48: filter_order = 8
            elif attenuation < 54: filter_order = 9
            elif attenuation < 60: filter_order = 10
            else: print "More than 10th order filter is required and out of scop
```

Figure 5: Determine Filter Order

The code as shown in Figure 6, Figure 7, and Figure 8 respectively show how the element values table is assigned which is maximally flat response, maximally flat time delay response, and equal ripple response (0.5dB)

```
#########################################################################
#
# Code below will assign prototype values
# According to the filter order calculated
# This is for Filter Response of: Max. Flat Response
#
#
#########################################################################

    if filter_response == 1:

        if filter_order == 10:

                a1=0.3129
                a2=0.9080
                a3=1.4142
                a4=1.7820
                a5=1.9754
                a6=1.9754
                a7=1.7820
                a8=1.4142
                a9=0.9080
                a10=0.3129
                a11=1.0000
```

Figure 6: Maximally Flat Response Element Values

```
#########################################################################
#
# Code below will assign prototype values
# According to the filter order calculated
# This is for Filter Response of: Max. Flat Time Delay Response
#
#
#########################################################################


      if filter_response == 2:

            if filter_order == 10:
                  a1=0.6305
                  a2=0.3002
                  a3=0.2384
                  a4=0.2066
                  a5=0.1808
                  a6=0.1539
                  a7=0.1240
                  a8=0.0911
                  a9=0.0557
                  a10=0.0187
                  a11=1.0000
```

Figure 7: Maximally Flat Time Delay Response E.ement Values

```
#########################################################################
#
# Code below will assign prototype values
# According to the filter order calculated
# This is for Filter Response of: Equal Ripple Response(0.5dB)
#
#
#########################################################################


      if filter_response == 3:

            if filter_order == 10:

                  a1=1.7543
                  a2=1.2721
                  a3=2.6754
                  a4=1.3725
                  a5=2.7392
                  a6=1.3806
                  a7=2.7231
                  a8=1.3485
                  a9=2.5239
                  a10=0.8842
                  a11=1.9841
```

Figure 8: Equal Ripple Response (0.5dB) Element Values

The code below as shown in Figure 9 will determine which filter to use, in this case it is low pass filter. It will do the low pass filter calculation with the appropriate filter order as to match the user specifications.

```
#########################################################################
#
# Code below will determine which filter to use
# Low pass, high pass, band pass, or band stop filter
# Also assign filter order needed
#
# LOW PASS FILTER
#########################################################################

      if filter_type == 1:
            if filter_order==1:
                  L1a = (output_resistance*a1*1000000000)/cutoff_frequency
                  G2a = a2*output_resistance

                  C1b = a1*1000000000000/(output_resistance*cutoff_frequency)
                  G2b = a2*output_resistance

                  print "\nType I Filter\n"
                  print "L1 = " ,L1a, " (nH) "
                  print "G2 = " ,G2a, " (Ohm) "

                  print "\nType II Filter\n"
                  print "C1 = " ,C1b, " (pF) "
                  print "G2 = " ,G2b, " (Ohm) "
```

Figure 9: Low Pass Filter Calculations

The code below as shown in Figure 10 will determine which filter to use, in this case it is high pass filter. It will do the low pass filter calculation with the appropriate filter order as to match the user specifications.

```
#########################################################################
#
# Code below will determine which filter to use
# Low pass, highpass, band pass, or band stop filter
# Also assign filter order needed
#
# HIGH PASS FILTER
#########################################################################


      elif filter_type ==2:
            if filter_order==1:

                  C1ax = 1000000000000/(output_resistance*cutoff_frequency*a1)
                  G2a = a2*output_resistance

                  L1bx = (output_resistance*1000000000)/(cutoff_frequency*a1)
                  G2b = a2*output_resistance

                  print "\nType I Filter\n"
                  print "C1 = " ,C1ax, " (pF) "
                  print "G2 = " ,G2a, " (Ohm) "


                  print "\nType II Filter\n"
                  print "L1 = " ,L1bx, " (nH) "
                  print "G2 = " ,G2b, " (Ohm) "
```

Figure 10: High Pass Filter Calculations

The code below as shown in Figure 11 will determine which filter to use, in this case it is band pass filter. It will do the low pass filter calculation with the appropriate filter order as to match the user specifications.

```
#########################################################################
#
# Code below will determine which filter to use
# Low pass, highpass, band pass, or band stop filter
# Also assign filter order needed
#
# BAND PASS FILTER
#########################################################################

      elif filter_type ==3:

            frequency_f_lower = input("Enter the Lower Frequency desired(Hz): "
            frequency_f_upper = input("Enter the Upper Frequency desired(Hz): "
            frequency_lower = 2*pi*frequency_f_lower
            frequency_upper = 2*pi*frequency_f_upper
            frequency_centre = sqrt(frequency_lower*frequency_upper)
            delta = (frequency_upper-frequency_lower)/frequency_centre

            if filter_order==1:

                  L1a = (a1*1000000000)/(frequency_centre*delta) #substitute
                  C1a = (delta*1000000000000)/(frequency_centre*a1) #substitu
                  G2a = a2*output_resistance

                  L1b = (delta*1000000000)/(frequency_centre*a1) #substitute
                  C1b = (a1*1000000000000)/(frequency_centre*delta) #substitu
                  G2b = a2*output_resistance

                  print "\nType I Filter\n"
                  print "L1 = " ,L1a, " (nH) "
                  print "C1 = " ,C1a, " (pF) "
                  print "G2 = " ,G2a, " (Ohm) "

                  print "\nType II Filter\n"
                  print "L1 = " ,L1b, " (nH) "
                  print "C1 = " ,C1b, " (pF) "
                  print "G2 = " ,G2b, " (Ohm) "
```

Figure 11: Band Pass Filter Calculations

The code below as shown in Figure 12 will determine which filter to use, in this case it is band stop filter. It will do the low pass filter calculation with the appropriate filter order as to match the user specifications.

```
#########################################################################
#
# Code below will determine which filter to use
# Low pass, highpass, band pass, or band stop filter
# Also assign filter order needed
#
# BAND STOP FILTER
#########################################################################

      elif filter_type ==4:

            frequency_f_lower = input("Enter the Lower Frequency desired(Hz): "
            frequency_f_upper = input("Enter the Upper Frequency desired(Hz): "
            frequency_lower = 2*pi*frequency_f_lower
            frequency_upper = 2*pi*frequency_f_upper
            frequency_centre = sqrt(frequency_lower*frequency_upper)
            delta = (frequency_upper-frequency_lower)/frequency_centre

            if filter_order==1:

                  L1a = (a1*delta*1000000000)/(frequency_centre) #substitute
                  C1a = (1000000000000)/(delta*frequency_centre*a1) #substitu
                  G2a = a2*output_resistance

                  L1b = (1000000000)/(delta*frequency_centre*a1) #substitute
                  C1b = (a1*delta*1000000000000)/(frequency_centre) #substitu
                  G2b = a2*output_resistance

                  print "\nType I Filter\n"
                  print "L1 = " ,L1a, " (nH) "
                  print "C1 = " ,C1a, " (pF) "
                  print "G2 = " ,G2a, " (Ohm) "

                  print "\nType II Filter\n"
                  print "L1 = " ,L1b, " (nH) "
                  print "C1 = " ,C1b, " (pF) "
                  print "G2 = " ,G2b, " (Ohm) "
```

Figure 12: Band Stop Filter Calculations

The code below as shown in Figure 13 will take input from the user for inductor value, capacitor value, and output resistance desired. After that, it will calculate the ABCD parameter of the two port network of low pass filter. First the ABCD parameter of the inductor is calculated. Then, it will calculate the ABCD parameter of the capacitor.

In two port network consisting of series inductor, the ABCD parameter of the two port network is A=1, B=j2πL, C=0, D=1. Whereas, in two port network consisting of shunt capacitor, the ABCD parameter of the two port network is A=1, B=0, C=1/(j2πC), D=1.

```
from numpy import *
from matplotlib.pyplot import *
from math import sqrt

######### Taking input from user

pi=3.141592654
L1a=input("Enter the Inductor, L1 value: ")
C2a=input("Enter the Capacitor, C2 value: ")
output_resistance=input("Enter the Output Resistance(ohm) desired: ")

####### 0.1GHZ freq

####### ABCD matrix calculation

A1_0_1 = 1
B1_0_1 = (1j*2*pi*0.1e12*L1a)
C1_0_1 = 0
D1_0_1 = 1

A2_0_1 = 1
B2_0_1 = 0
C2_0_1 = 1/(1j*2*pi*0.1e12*C2a)
D2_0_1 = 1
```

Figure 13: ABCD Parameter of the Low Pass Filter

The ABCD matrix of the inductor is multiplied by the ABCD matrix of the capacitor. This is done because the two port circuit is cascaded. Then, ABCD parameter of the system is converted to S parameter. The amplitude of S12 is derived from here.

```
####### ABCD parameter for the system cascaded

W_0_1=(A1_0_1*A2_0_1)+(B1_0_1*C2_0_1)
X_0_1=(A1_0_1*B2_0_1)+(B1_0_1*D2_0_1)
Y_0_1=(C1_0_1*A2_0_1)+(D1_0_1*C2_0_1)
Z_0_1=(C1_0_1*B2_0_1)+(D1_0_1*D2_0_1)

####### ABCD parameter is then converted to S parameter

S11_0_1 = (W_0_1 + (X_0_1/output_resistance) - (Y_0_1*output_resistance) - Z_0_1
S12_0_1 = (2*((W_0_1*Z_0_1)-(X_0_1*Y_0_1)))/(W_0_1 + (X_0_1/output_resistance) +
S21_0_1 = 2/(W_0_1 + (X_0_1/output_resistance) + (Y_0_1*output_resistance) + Z_0_
S22_0_1 = (- W_0_1 + (X_0_1/output_resistance) - (Y_0_1*output_resistance) + Z_0_

S12amp_0_1=sqrt((S12_0_1.real*S12_0_1.real)+(S12_0_1.imag*S12_0_1.imag))
```

Figure 14: ABCD Parameter to S Parameters Conversion

Finally, a graph is created. It will account for the frequency between 0.1 GHz to 10 GHz. the S12 amplitude is plotted on the graph in the y-axis with their corresponding frequency value in the x-axis. This will show the filter response of the system.

```
########## plotting the graph

figure()
y=array([S12amp_0_1, S12amp_1, S12amp_2, S12amp_3, S12amp_4, S12amp_5, S12amp_6
t=array([0.1,1,2,3,4,5,6,7,8,9,10])
plot(t,y, 'x:')
show()
```

Figure 15: Plotting Filter Response Graph

4.3 Python Code Examples

For this part, here are some examples of the actual running code. The source code module is compiled and it will run on IDLE. The user has the ability to enter the frequency, cutoff frequency, attenuation, and output resistance desired. For the following example the low pass filter is chosen and the maximally flat filter response is also chosen.

```
IDLE 2.6.5      ==== No Subprocess ====
>>>
Enter the Frequency desired(Hz): 2e9
Enter the Cutoff Frequency desired(Hz): 1e9
Enter the Attenuation(dB) desired: 15
Enter the Output Resistance(ohm) desired: 50
Enter which filter type [1],[2],[3],or [4] is desired:

[1]Low Pass
[2]High Pass
[3]Band Pass
[4]Band Stop

 1

Enter which filter response [1],[2],or [3] is desired:

[1]Max. Flat
[2]Max. Flay Time Delay
[3]Equal Ripple

 1
```

Figure 16: Input Part of Low Pass Filter

After the input has been entered, the program will calculate the normalized frequency and assign filter order automatically depending on the attenuation vs. normalized frequency graph. In the following example, it assigns filter order of 4. Therefore, the output will show 4 lumped elements consisting of inductor and capacitor. The output will also provide the user for Type I and Type II filter. The reason behind this is because the user can determine which lumped elements are more suitable and compare them between the two.

```
Type I Filter

L1 =   6.09085967133   (nH)
C2 =   5.88173007614   (pF)
L3 =   14.7043251903   (nH)
C4 =   2.43634386853   (pF)
G5 =   50.0   (Ohm)

Type II Filter

C1 =   2.43634386853   (pF)
L2 =   14.7043251903   (nH)
C3 =   5.88173007614   (pF)
L4 =   6.09085967133   (nH)
G5 =   50.0   (Ohm)
>>>
```

Figure 17: Output Part of Low Pass Filter

The next program will require the user to enter the inductor, L1 value and capacitor, C2 value. This is because the filter used here is the second order filter. Output resistance is specified. On the background, the program will calculate the ABCD matrix of the two port network for both the inductor and capacitor. This two ABCD matrix is multiplied together because the two port network is cascaded. Then, the ABCD matrix of the system is known and is converted to the S parameter. This routine will be repeated for the frequency range between 0.1 GHz until 10 GHz.

```
IDLE 2.6.5       ==== No Subprocess ====
>>>
Enter the Inductor, L1 value: 1e-9
Enter the Capacitor, C2 value: 2e-12
Enter the Output Resistance(ohm) desired: 50
```

Figure 18: Input Part of Plotting Program

When the S12 values are retrieved, the values are in the form of imaginary numbers. Then, the S12 values are converted to complex numbers and the amplitude is captured. The graph of amplitude S12 vs. frequency is plotted as shown below. This program will plot the graph between the frequencies of 0.1 GHz to 10 GHz. The y-axis refers to the amplitude of S12 and the x-axis refers to the frequency range between 0.1 GHz to 10 GHz. The graph shown below is the Amplitude S12 vs. Frequency Graph.

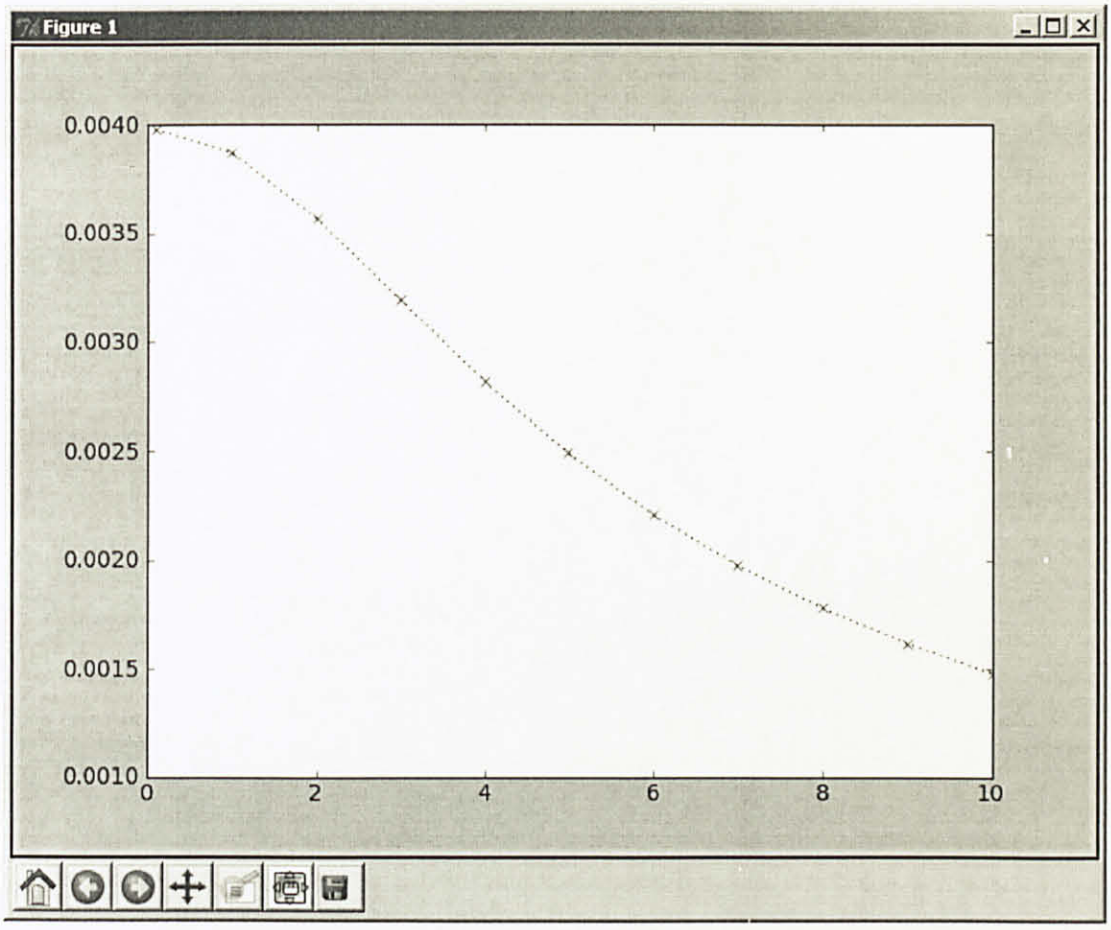

Figure 19: Output Part of Plotting Program

CHAPTER 5
CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The program for the filter design by the insertion loss method has been developed as shown by the code in Figure 4 until Figure 19. The filter transformation capability is developed which will enable the use of high pass, band pass, and band stop filter using the low pass filter reactive elements as the foundation. This calculation will make use of the filter transformation and filter implementation. It is the most accurate and reliable source which can be used in helping to design a microwave filter circuit. The filter response can be viewed fast enough from the plotted graph to get the rough ideas about the filter performance.

5.2 Recommendations

This program need to be expanded more to cover the filter implementation. This includes Impedance and Admittance Inverters and others.

REFERENCES

[1] David M. Pozar, *Microwave Engineering,* New York, John Wiley & Sons

[2] Python Code Tutorial, 10 Feb 2010 <http://www.python.org/> 4 April 2010

[3] Tony Gaddis, *Starting Out With Python*, Pearson

APPENDICES

APPENDIX A

MICROWAVE FILTER COMPUTER CODE

```
#########################################################################
#
#
#
#
#
#
#########################################################################

#########################################################################
#
# Main function definition
#
#
#
#
#########################################################################

from math import sqrt

frequency_f =input("Enter the Frequency desired(Hz): ")
cutoff_frequency_f =input("Enter the Cutoff Frequency desired(Hz): ")
attenuation = input("Enter the Attenuation(dB) desired: ")

pi=3.141592654

frequency = 2*pi*frequency_f
cutoff_frequency = 2*pi*cutoff_frequency_f

normalized_frequency= abs(frequency / cutoff_frequency) - 1

output_resistance =input("Enter the Output Resistance(ohm) desired: ")
filter_type =input("Enter which filter type [1],[2],[3],or [4] is desired:\n\n[1]Low Pass\n[2]High
Pass\n[3]Band Pass\n[4]Band Stop\n\n ")
filter_response =input("\nEnter which filter response [1],[2],or [3] is desired:\n\n[1]Max.
Flat\n[2]Max. Flay Time Delay\n[3]Equal Ripple\n\n ")


#########################################################################
#
```

```
# Code below will determine the filter order
# based on the frequency, cutoff frequency, and attenuation entered
#
#
#
#########################################################################
if pi == 3.141592654:

    if normalized_frequency > 1:
        if attenuation < 7: filter_order = 1
        elif attenuation < 12: filter_order = 2
        elif attenuation < 18: filter_order = 3
        elif attenuation < 24: filter_order = 4
        elif attenuation < 30: filter_order = 5
        elif attenuation < 37: filter_order = 6
        elif attenuation < 42: filter_order = 7
        elif attenuation < 48: filter_order = 8
        elif attenuation < 54: filter_order = 9
        elif attenuation < 60: filter_order = 10
        else: print "More than 10th order filter is required and out of scope of this program"

    elif normalized_frequency > 0.7:

        if attenuation < 6: filter_order = 1
        elif attenuation < 10: filter_order = 2
        elif attenuation < 14: filter_order = 3
        elif attenuation < 18: filter_order = 4
        elif attenuation < 23: filter_order = 5
        elif attenuation < 28: filter_order = 6
        elif attenuation < 32: filter_order = 7
        elif attenuation < 36: filter_order = 8
        elif attenuation < 40: filter_order = 9
        elif attenuation < 44: filter_order = 10
        else: print "More than 10th order filter is required and out of scope of this program"

    elif normalized_frequency > 0.5:
        if attenuation < 6: filter_order = 1
        elif attenuation < 8: filter_order = 2
        elif attenuation < 11: filter_order = 3
        elif attenuation < 14: filter_order = 4
        elif attenuation < 18: filter_order = 5
        elif attenuation < 22: filter_order = 6
        elif attenuation < 24: filter_order = 7
        elif attenuation < 28: filter_order = 8
        elif attenuation < 31: filter_order = 9
        elif attenuation < 34: filter_order = 10
        else: print "More than 10th order filter is required and out of scope of this program"

    elif normalized_frequency > 0.3:
        if attenuation < 4: filter_order = 1
        elif attenuation < 6: filter_order = 2
        elif attenuation < 8: filter_order = 3
        elif attenuation < 9: filter_order = 4
        elif attenuation < 12: filter_order = 5
```

```
        elif attenuation < 15: filter_order = 6
        elif attenuation < 16: filter_order = 7
        elif attenuation < 18: filter_order = 8
        elif attenuation < 20: filter_order = 9
        elif attenuation < 21: filter_order = 10
        else: print "More than 10th order filter is required and out of scope of this program"

    elif normalized_frequency > 0.2:
        if attenuation < 4: filter_order = 1
        elif attenuation < 5: filter_order = 2
        elif attenuation < 6: filter_order = 3
        elif attenuation < 7: filter_order = 4
        elif attenuation < 8: filter_order = 5
        elif attenuation < 10: filter_order = 6
        elif attenuation < 12: filter_order = 7
        elif attenuation < 14: filter_order = 8
        elif attenuation < 15: filter_order = 9
        elif attenuation < 16: filter_order = 10
        else: print "More than 10th order filter is required and out of scope of this program"

    elif normalized_frequency > 0.1:
        if attenuation < 3: filter_order = 1
        elif attenuation < 4: filter_order = 2
        elif attenuation < 5: filter_order = 3
        elif attenuation < 6: filter_order = 4
        elif attenuation < 7: filter_order = 5
        elif attenuation < 7.5: filter_order = 6
        elif attenuation < 8: filter_order = 7
        elif attenuation < 8.5: filter_order = 8
        elif attenuation < 9: filter_order = 9
        elif attenuation < 9.5: filter_order = 10
        else: print "More than 10th order filter is required and out of scope of this program"

    else: print "The value of normalized frequency required is too low (less than 0.1) and out of scope
of this program"

#######################################################################
#
# Code below will assign prototype values
# According to the filter order calculated
# This is for Filter Response of: Max. Flat Response
#
#
#######################################################################


    if filter_response == 1:

        if filter_order == 10:

            a1=0.3129
            a2=0.9080
            a3=1.4142
            a4=1.7820
```

```
        a5=1.9754
        a6=1.9754
        a7=1.7820
        a8=1.4142
        a9=0.9080
        a10=0.3129
        a11=1.0000

    elif filter_order == 9:
        a1=0.3473
        a2=1.0000
        a3=1.5321
        a4=1.8794
        a5=2.0000
        a6=1.8794
        a7=1.5321
        a8=1.0000
        a9=0.3473
        a10=1.0000

    elif filter_order == 8:
        a1=0.3902
        a2=1.1111
        a3=1.6629
        a4=1.9615
        a5=1.9615
        a6=1.6629
        a7=1.1111
        a8=0.3902
        a9=1.0000

    elif filter_order == 7:
        a1=0.4450
        a2=1.2470
        a3=1.8019
        a4=2.0000
        a5=1.8019
        a6=1.2470
        a7=0.4450
        a8=1.0000

    elif filter_order == 6:
        a1=0.5176
        a2=1.4142
        a3=1.9318
        a4=1.9318
        a5=1.4142
        a6=0.5176
        a7=1.0000

    elif filter_order == 5:
        a1=0.6180
        a2=1.6180
        a3=2.0000
```

```
            a4=1.6180
            a5=0.1680
            a6=1.0000

        elif filter_order == 4:
            a1=0.7654
            a2=1.8478
            a3=1.8478
            a4=0.7654
            a5=1.0000

        elif filter_order == 3:
            a1=1.0000
            a2=2.0000
            a3=1.0000
            a4=1.0000

        elif filter_order == 2:
            a1=1.4142
            a2=1.4142
            a3=1.0000

        elif filter_order == 1:
            a1=2.0000
            a2=1.0000

        else:
            print "Please enter filter order between 1 to 10"


#########################################################################
#
# Code below will assign prototype values
# According to the filter order calculated
# This is for Filter Response of: Max. Flat Time Delay Response
#
#
#########################################################################


    if filter_response == 2:

        if filter_order == 10:
            a1=0.6305
            a2=0.3002
            a3=0.2384
            a4=0.2066
            a5=0.1808
            a6=0.1539
            a7=0.1240
            a8=0.0911
            a9=0.0557
            a10=0.0187
            a11=1.0000
```

```
elif filter_order == 9:
        a1=0.6678
        a2=0.3023
        a3=0.2547
        a4=0.2184
        a5=0.1859
        a6=0.1506
        a7=0.1111
        a8=0.0682
        a9=0.0230
        a10=1.0000

elif filter_order == 8:
        a1=0.7125
        a2=0.3446
        a3=0.2735
        a4=0.2297
        a5=0.1867
        a6=0.1387
        a7=0.0855
        a8=0.0289
        a9=1.0000

elif filter_order == 7:
        a1=0.7677
        a2=0.3744
        a3=0.2944
        a4=0.2378
        a5=0.1778
        a6=0.1104
        a7=0.0375
        a8=1.0000

elif filter_order == 6:
        a1=0.8377
        a2=0.4116
        a3=0.3158
        a4=0.2364
        a5=0.1480
        a6=0.0505
        a7=1.0000

elif filter_order == 5:

        a1=0.9303
        a2=0.4577
        a3=0.3312
        a4=0.2090
        a5=0.0718
        a6=1.0000

elif filter_order == 4:
```

```
            a1=1.0598
            a2=0.5116
            a3=0.3181
            a4=0.1104
            a5=1.0000

        elif filter_order == 3:

            a1=1.2550
            a2=0.5528
            a3=0.1922
            a4=1.0000

        elif filter_order == 2:

            a1=1.5774
            a2=0.4226
            a3=1.0000

        elif filter_order == 1:

            a1=2.0000
            a2=1.0000

        else:
            print "Please enter filter order between 1 to 10"


#########################################################################
#
# Code below will assign prototype values
# According to the filter order calculated
# This is for Filter Response of: Equal Ripple Response(0.5dB)
#
#
#########################################################################


    if filter_response == 3:

        if filter_order == 10:

            a1=1.7543
            a2=1.2721
            a3=2.6754
            a4=1.3725
            a5=2.7392
            a6=1.3806
            a7=2.7231
            a8=1.3485
            a9=2.5239
            a10=0.8842
            a11=1.9841
```

```
elif filter_order == 9:

    a1=1.7504
    a2=1.2690
    a3=2.6678
    a4=1.3673
    a5=2.7239
    a6=1.3673
    a7=2.6678
    a8=1.2690
    a9=1.7504
    a10=1.0000

elif filter_order == 8:

    a1=1.7451
    a2=1.2647
    a3=2.6564
    a4=1.3590
    a5=2.6964
    a6=1.3389
    a7=2.5093
    a8=0.8796
    a9=1.9841

elif filter_order == 7:

    a1=1.7372
    a2=1.2583
    a3=2.6381
    a4=1.3444
    a5=2.6381
    a6=1.2583
    a7=1.7372
    a8=1.0000

elif filter_order == 6:

    a1=1.7254
    a2=1.2479
    a3=2.6064
    a4=1.3137
    a5=2.4758
    a6=0.8696
    a7=1.9841

elif filter_order == 5:

    a1=1.7058
    a2=1.2296
    a3=2.5408
    a4=1.2296
    a5=1.7058
```

```
            a6=1.0000

        elif filter_order == 4:

            a1=1.6703
            a2=1.1926
            a3=2.3661
            a4=0.8419
            a5=1.9841

        elif filter_order == 3:

            a1=1.5963
            a2=1.0967
            a3=1.5963
            a4=1.0000

        elif filter_order == 2:

            a1=1.4029
            a2=0.7071
            a3=1.9841

        elif filter_order == 1:

            a1=0.6986
            a2=1.0000

        else:
            print "Please enter filter order between 1 to 10"

# else:
# print "Please enter filter response between 1 to 3"


#########################################################################
#
# Code below will determine which filter to use
# Low pass, high pass, band pass, or band stop filter
# Also assign filter order needed
#
# LOW PASS FILTER
#########################################################################

    if filter_type == 1:
        if filter_order==1:
            L1a = (output_resistance*a1*1000000000)/cutoff_frequency
            G2a = a2*output_resistance

            C1b = a1*1000000000000/(output_resistance*cutoff_frequency)
            G2b = a2*output_resistance

            print "\nType I Filter\n"
            print "L1 = " ,L1a, " (nH) "
```

```
        print "G2 = " ,G2a, " (Ohm) "

        print "\nType II Filter\n"
        print "C1 = " ,C1b, " (pF) "
        print "G2 = " ,G2b, " (Ohm) "

elif filter_order==2:

        L1a = (output_resistance*a1*1000000000)/cutoff_frequency
        C2a = a2*1000000000000/(output_resistance*cutoff_frequency)
        G3a = a3*output_resistance

        C1b = a1*1000000000000/(output_resistance*cutoff_frequency)
        L2b = (output_resistance*a2*1000000000)/cutoff_frequency
        G3b = a3*output_resistance

        print "\nType I Filter\n"
        print "L1 = " ,L1a, " (nH) "
        print "C2 = " ,C2a, " (pF) "
        print "G3 = " ,G3a, " (Ohm) "

        print "\nType II Filter\n"
        print "C1 = " ,C1b, " (pF) "
        print "L2 = " ,L2b, " (nH) "
        print "G3 = " ,G3b, " (Ohm) "

elif filter_order==3:

        L1a = (output_resistance*a1*1000000000)/cutoff_frequency
        C2a = a2*1000000000000/(output_resistance*cutoff_frequency)
        L3a = (output_resistance*a3*1000000000)/cutoff_frequency
        G4a = a4*output_resistance

        C1b = a1*1000000000000/(output_resistance*cutoff_frequency)
        L2b = (output_resistance*a2*1000000000)/cutoff_frequency
        C3b = a3*1000000000000/(output_resistance*cutoff_frequency)
        G4b = a4*output_resistance

        print "\nType I Filter\n"
        print "L1 = " ,L1a, " (nH) "
        print "C2 = " ,C2a, " (pF) "
        print "L3 = " ,L3a, " (nH) "
        print "G4 = " ,G4a, " (Ohm) "

        print "\nType II Filter\n"
        print "C1 = " ,C1b, " (pF) "
        print "L2 = " ,L2b, " (nH) "
        print "C3 = " ,C3b, " (pF) "
        print "G4 = " ,G4b, " (Ohm) "

elif filter_order==4:

        L1a = (output_resistance*a1*1000000000)/cutoff_frequency
        C2a = a2*1000000000000/(output_resistance*cutoff_frequency)
```

```
        L3a = (output_resistance*a3*1000000000)/cutoff_frequency
        C4a = a4*1000000000000/(output_resistance*cutoff_frequency)
        G5a = a5*output_resistance

        C1b = a1*1000000000000/(output_resistance*cutoff_frequency)
        L2b = (output_resistance*a2*1000000000)/cutoff_frequency
        C3b = a3*1000000000000/(output_resistance*cutoff_frequency)
        L4b = (output_resistance*a4*1000000000)/cutoff_frequency
        G5b = a5*output_resistance

        print "\nType I Filter\n"
        print "L1 = " ,L1a, " (nH) "
        print "C2 = " ,C2a, " (pF) "
        print "L3 = " ,L3a, " (nH) "
        print "C4 = " ,C4a, " (pF) "
        print "G5 = " ,G5a, " (Ohm) "

        print "\nType II Filter\n"
        print "C1 = " ,C1b, " (pF) "
        print "L2 = " ,L2b, " (nH) "
        print "C3 = " ,C3b, " (pF) "
        print "L4 = " ,L4b, " (nH) "
        print "G5 = " ,G5b, " (Ohm) "

elif filter_order==5:

        L1a = (output_resistance*a1*1000000000)/cutoff_frequency
        C2a = a2*1000000000000/(output_resistance*cutoff_frequency)
        L3a = (output_resistance*a3*1000000000)/cutoff_frequency
        C4a = a4*1000000000000/(output_resistance*cutoff_frequency)
        L5a = (output_resistance*a5*1000000000)/cutoff_frequency
        G6a = a6*output_resistance

        C1b = a1*1000000000000/(output_resistance*cutoff_frequency)
        L2b = (output_resistance*a2*1000000000)/cutoff_frequency
        C3b = a3*1000000000000/(output_resistance*cutoff_frequency)
        L4b = (output_resistance*a4*1000000000)/cutoff_frequency
        C5b = a5*1000000000000/(output_resistance*cutoff_frequency)
        G6b = a6*output_resistance

        print "\nType I Filter\n"
        print "L1 = " ,L1a, " (nH) "
        print "C2 = " ,C2a, " (pF) "
        print "L3 = " ,L3a, " (nH) "
        print "C4 = " ,C4a, " (pF) "
        print "L5 = " ,L5a, " (nH) "
        print "G6 = " ,G6a, " (Ohm) "

        print "\nType II Filter\n"
        print "C1 = " ,C1b, " (pF) "
        print "L2 = " ,L2b, " (nH) "
        print "C3 = " ,C3b, " (pF) "
        print "L4 = " ,L4b, " (nH) "
        print "C5 = " ,C5b, " (pF) "
```

```
        print "G6 = " ,G6b, " (Ohm) "

    elif filter_order==6:

        L1a = (output_resistance*a1*1000000000)/cutoff_frequency
        C2a = a2*1000000000000/(output_resistance*cutoff_frequency)
        L3a = (output_resistance*a3*1000000000)/cutoff_frequency
        C4a = a4*1000000000000/(output_resistance*cutoff_frequency)
        L5a = (output_resistance*a5*1000000000)/cutoff_frequency
        C6a = a6*1000000000000/(output_resistance*cutoff_frequency)
        G7a = a7*output_resistance

        C1b = a1*1000000000000/(output_resistance*cutoff_frequency)
        L2b = (output_resistance*a2*1000000000)/cutoff_frequency
        C3b = a3*1000000000000/(output_resistance*cutoff_frequency)
        L4b = (output_resistance*a4*1000000000)/cutoff_frequency
        C5b = a5*1000000000000/(output_resistance*cutoff_frequency)
        L6b = (output_resistance*a6*1000000000)/cutoff_frequency
        G7b = a7*output_resistance

        print "\nType I Filter\n"
        print "L1 = " ,L1a, " (nH) "
        print "C2 = " ,C2a, " (pF) "
        print "L3 = " ,L3a, " (nH) "
        print "C4 = " ,C4a, " (pF) "
        print "L5 = " ,L5a, " (nH) "
        print "C6 = " ,C6a, " (pF) "
        print "G7 = " ,G7a, " (Ohm) "

        print "\nType II Filter\n"
        print "C1 = " ,C1b, " (pF) "
        print "L2 = " ,L2b, " (nH) "
        print "C3 = " ,C3b, " (pF) "
        print "L4 = " ,L4b, " (nH) "
        print "C5 = " ,C5b, " (pF) "
        print "L6 = " ,L6b, " (nH) "
        print "G7 = " ,G7b, " (Ohm) "

elif filter_order==7:

        L1a = (output_resistance*a1*1000000000)/cutoff_frequency
        C2a = a2*1000000000000/(output_resistance*cutoff_frequency)
        L3a = (output_resistance*a3*1000000000)/cutoff_frequency
        C4a = a4*1000000000000/(output_resistance*cutoff_frequency)
        L5a = (output_resistance*a5*1000000000)/cutoff_frequency
        C6a = a6*1000000000000/(output_resistance*cutoff_frequency)
        L7a = (output_resistance*a7*1000000000)/cutoff_frequency
        G8a = a8*output_resistance

        C1b = a1*1000000000000/(output_resistance*cutoff_frequency)
        L2b = (output_resistance*a2*1000000000)/cutoff_frequency
        C3b = a3*1000000000000/(output_resistance*cutoff_frequency)
        L4b = (output_resistance*a4*1000000000)/cutoff_frequency
        C5b = a5*1000000000000/(output_resistance*cutoff_frequency)
```

```
        L6b = (output_resistance*a6*1000000000)/cutoff_frequency
        C7b = a7*1000000000000/(output_resistance*cutoff_frequency)
        G8b = a8*output_resistance

        print "\nType I Filter\n"
        print "L1 = " ,L1a, " (nH) "
        print "C2 = " ,C2a, " (pF) "
        print "L3 = " ,L3a, " (nH) "
        print "C4 = " ,C4a, " (pF) "
        print "L5 = " ,L5a, " (nH) "
        print "C6 = " ,C6a, " (pF) "
        print "L7 = " ,L7a, " (nH) "
        print "C8 = " ,C8a, " (pF) "
        print "G8 = " ,G8a, " (Ohm) "

        print "\nType II Filter\n"
        print "C1 = " ,C1b, " (pF) "
        print "L2 = " ,L2b, " (nH) "
        print "C3 = " ,C3b, " (pF) "
        print "L4 = " ,L4b, " (nH) "
        print "C5 = " ,C5b, " (pF) "
        print "L6 = " ,L6b, " (nH) "
        print "C7 = " ,C7b, " (pF) "
        print "L8 = " ,L8b, " (nH) "
        print "G8 = " ,G8b, " (Ohm) "

elif filter_order==8:

        L1a = (output_resistance*a1*1000000000)/cutoff_frequency
        C2a = a2*1000000000000/(output_resistance*cutoff_frequency)
        L3a = (output_resistance*a3*1000000000)/cutoff_frequency
        C4a = a4*1000000000000/(output_resistance*cutoff_frequency)
        L5a = (output_resistance*a5*1000000000)/cutoff_frequency
        C6a = a6*1000000000000/(output_resistance*cutoff_frequency)
        L7a = (output_resistance*a7*1000000000)/cutoff_frequency
        C8a = a8*1000000000000/(output_resistance*cutoff_frequency)
        G9a = a9*output_resistance

        C1b = a1*1000000000000/(output_resistance*cutoff_frequency)
        L2b = (output_resistance*a2*1000000000)/cutoff_frequency
        C3b = a3*1000000000000/(output_resistance*cutoff_frequency)
        L4b = (output_resistance*a4*1000000000)/cutoff_frequency
        C5b = a5*1000000000000/(output_resistance*cutoff_frequency)
        L6b = (output_resistance*a6*1000000000)/cutoff_frequency
        C7b = a7*1000000000000/(output_resistance*cutoff_frequency)
        L8b = (output_resistance*a8*1000000000)/cutoff_frequency
        G9b = a9*output_resistance

        print "\nType I Filter\n"
        print "L1 = " ,L1a, " (nH) "
        print "C2 = " ,C2a, " (pF) "
        print "L3 = " ,L3a, " (nH) "
        print "C4 = " ,C4a, " (pF) "
        print "L5 = " ,L5a, " (nH) "
```

```
        print "C6 = " ,C6a, " (pF) "
        print "L7 = " ,L7a, " (nH) "
        print "C8 = " ,C8a, " (pF) "
        print "G9 = " ,G9a, " (Ohm) "

        print "\nType II Filter\n"
        print "C1 = " ,C1b, " (pF) "
        print "L2 = " ,L2b, " (nH) "
        print "C3 = " ,C3b, " (pF) "
        print "L4 = " ,L4b, " (nH) "
        print "C5 = " ,C5b, " (pF) "
        print "L6 = " ,L6b, " (nH) "
        print "C7 = " ,C7b, " (pF) "
        print "L8 = " ,L8b, " (nH) "
        print "G9 = " ,G9b, " (Ohm) "

elif filter_order==9:

        L1a = (output_resistance*a1*1000000000)/cutoff_frequency
        C2a = a2*1000000000000/(output_resistance*cutoff_frequency)
        L3a = (output_resistance*a3*1000000000)/cutoff_frequency
        C4a = a4*1000000000000/(output_resistance*cutoff_frequency)
        L5a = (output_resistance*a5*1000000000)/cutoff_frequency
        C6a = a6*1000000000000/(output_resistance*cutoff_frequency)
        L7a = (output_resistance*a7*1000000000)/cutoff_frequency
        C8a = a8*1000000000000/(output_resistance*cutoff_frequency)
        L9a = (output_resistance*a9*1000000000)/cutoff_frequency
        G10a = a10*output_resistance

        C1b = a1*1000000000000/(output_resistance*cutoff_frequency)
        L2b = (output_resistance*a2*1000000000)/cutoff_frequency
        C3b = a3*1000000000000/(output_resistance*cutoff_frequency)
        L4b = (output_resistance*a4*1000000000)/cutoff_frequency
        C5b = a5*1000000000000/(output_resistance*cutoff_frequency)
        L6b = (output_resistance*a6*1000000000)/cutoff_frequency
        C7b = a7*1000000000000/(output_resistance*cutoff_frequency)
        L8b = (output_resistance*a8*1000000000)/cutoff_frequency
        C9b = a9*1000000000000/(output_resistance*cutoff_frequency)
        G10b = a10*output_resistance

        print "\nType I Filter\n"
        print "L1 = " ,L1a, " (nH) "
        print "C2 = " ,C2a, " (pF) "
        print "L3 = " ,L3a, " (nH) "
        print "C4 = " ,C4a, " (pF) "
        print "L5 = " ,L5a, " (nH) "
        print "C6 = " ,C6a, " (pF) "
        print "L7 = " ,L7a, " (nH) "
        print "C8 = " ,C8a, " (pF) "
        print "L9 = " ,L9a, " (nH) "
        print "G10 = " ,G10a, " (Ohm) "
```

```
        print "\nType II Filter\n"
        print "C1 = " ,C1b, " (pF) "
        print "L2 = " ,L2b, " (nH) "
        print "C3 = " ,C3b, " (pF) "
        print "L4 = " ,L4b, " (nH) "
        print "C5 = " ,C5b, " (pF) "
        print "L6 = " ,L6b, " (nH) "
        print "C7 = " ,C7b, " (pF) "
        print "L8 = " ,L8b, " (nH) "
        print "C9 = " ,C9b, " (pF) "
        print "G10= " ,G10b, " (Ohm) "


elif filter_order==10:

        L1a = (output_resistance*a1*1000000000)/cutoff_frequency
        C2a = a2*1000000000000/(output_resistance*cutoff_frequency)
        L3a = (output_resistance*a3*1000000000)/cutoff_frequency
        C4a = a4*1000000000000/(output_resistance*cutoff_frequency)
        L5a = (output_resistance*a5*1000000000)/cutoff_frequency
        C6a = a6*1000000000000/(output_resistance*cutoff_frequency)
        L7a = (output_resistance*a7*1000000000)/cutoff_frequency
        C8a = a8*1000000000000/(output_resistance*cutoff_frequency)
        L9a = (output_resistance*a9*1000000000)/cutoff_frequency
        C10a = a10*1000000000000/(output_resistance*cutoff_frequency)
        G11a = a11*output_resistance

        C1b = a1*1000000000000/(output_resistance*cutoff_frequency)
        L2b = (output_resistance*a2*1000000000)/cutoff_frequency
        C3b = a3*1000000000000/(output_resistance*cutoff_frequency)
        L4b = (output_resistance*a4*1000000000)/cutoff_frequency
        C5b = a5*1000000000000/(output_resistance*cutoff_frequency)
        L6b = (output_resistance*a6*1000000000)/cutoff_frequency
        C7b = a7*1000000000000/(output_resistance*cutoff_frequency)
        L8b = (output_resistance*a8*1000000000)/cutoff_frequency
        C9b = a9*1000000000000/(output_resistance*cutoff_frequency)
        L10b = (output_resistance*a10*1000000000)/cutoff_frequency
        G11b = a11*output_resistance

        print "\nType I Filter\n"
        print "L1 = " ,L1a, " (nH) "
        print "C2 = " ,C2a, " (pF) "
        print "L3 = " ,L3a, " (nH) "
        print "C4 = " ,C4a, " (pF) "
        print "L5 = " ,L5a, " (nH) "
        print "C6 = " ,C6a, " (pF) "
        print "L7 = " ,L7a, " (nH) "
        print "C8 = " ,C8a, " (pF) "
        print "L9 = " ,L9a, " (nH) "
        print "C10 = " ,C10a, " (pF) "
        print "G11 = " ,G11a, " (Ohm) "

        print "\nType II Filter\n"
        print "C1 = " ,C1b, " (pF) "
```

```
        print "\nType I Filter\n"
        print "C1 = " ,C1ax, " (pF) "
        print "L2 = " ,L2ax, " (nH) "
        print "G3 = " ,G3a, " (Ohm) "


        print "\nType II Filter\n"
        print "L1 = " ,L1bx, " (nH) "
        print "C2 = " ,C2bx, " (pF) "
        print "G3 = " ,G3b, " (Ohm) "

elif filter_order==3:

        C1ax = 1000000000000/(output_resistance*cutoff_frequency*a1)
        L2ax = (output_resistance*1000000000)/(cutoff_frequency*a2)
        C3ax = 1000000000000/(output_resistance*cutoff_frequency*a3)
        G4a = a4*output_resistance

        L1bx = (output_resistance*1000000000)/(cutoff_frequer cy*a1)
        C2bx = 1000000000000/(output_resistance*cutoff_frequency*a2)
        L3bx = (output_resistance*1000000000)/(cutoff_frequency*a3)
        G4b = a4*output_resistance

        print "\nType I Filter\n"
        print "C1 = " ,C1ax, " (pF) "
        print "L2 = " ,L2ax, " (nH) "
        print "C3 = " ,C3ax, " (pF) "
        print "G4 = " ,G4a, " (Ohm) "


        print "\nType II Filter\n"
        print "L1 = " ,L1bx, " (nH) "
        print "C2 = " ,C2bx, " (pF) "
        print "L3 = " ,L3bx, " (nH) "
        print "G4 = " ,G4b, " (Ohm) "

elif filter_order==4:

        C1ax = 1000000000000/(output_resistance*cutoff_frequency*a1)
        L2ax = (output_resistance*1000000000)/(cutoff_frequency*a2)
        C3ax = 1000000000000/(output_resistance*cutoff_frequency*a3)
        L4ax = (output_resistance*1000000000)/(cutoff_frequency*a4)
        G5a = a5*output_resistance

        L1bx = (output_resistance*1000000000)/(cutoff_frequency*a1)
        C2bx = 1000000000000/(output_resistance*cutoff_frequency*a2)
        L3bx = (output_resistance*1000000000)/(cutoff_frequency*a3)
        C4bx = 1000000000000/(output_resistance*cutoff_frequency*a4)
        G5b = a5*output_resistance

        print "\nType I Filter\n"
        print "C1 = " ,C1ax, " (pF) "
        print "L2 = " ,L2ax, " (nH) "
        print "C3 = " ,C3ax, " (pF) "
```

```
        print "L4 = " ,L4ax, " (nH) "
        print "G5 = " ,G5a, " (Ohm) "


        print "\nType II Filter\n"
        print "L1 = " ,L1bx, " (nH) "
        print "C2 = " ,C2bx, " (pF) "
        print "L3 = " ,L3bx, " (nH) "
        print "C4 = " ,C4bx, " (pF) "
        print "G5 = " ,G5b, " (Ohm) "

elif filter_order==5:

        C1ax = 1000000000000/(output_resistance*cutoff_frequency*a1)
        L2ax = (output_resistance*1000000000)/(cutoff_frequency*a2)
        C3ax = 1000000000000/(output_resistance*cutoff_frequency*a3)
        L4ax = (output_resistance*1000000000)/(cutoff_frequency*a4)
        C5ax = 1000000000000/(output_resistance*cutoff_frequency*a5)
        G6a = a6*output_resistance

        L1bx = (output_resistance*1000000000)/(cutoff_frequency*a1)
        C2bx = 1000000000000/(output_resistance*cutoff_frequency*a2)
        L3bx = (output_resistance*1000000000)/(cutoff_frequency*a3)
        C4bx = 1000000000000/(output_resistance*cutoff_frequency*a4)
        L5bx = (output_resistance*1000000000)/(cutoff_frequency*a5)
        G6b = a6*output_resistance

        print "\nType I Filter\n"
        print "C1 = " ,C1ax, " (pF) "
        print "L2 = " ,L2ax, " (nH) "
        print "C3 = " ,C3ax, " (pF) "
        print "L4 = " ,L4ax, " (nH) "
        print "C5 = " ,C5ax, " (pF) "
        print "G6 = " ,G6a, " (Ohm) "


        print "\nType II Filter\n"
        print "L1 = " ,L1bx, " (nH) "
        print "C2 = " ,C2bx, " (pF) "
        print "L3 = " ,L3bx, " (nH) "
        print "C4 = " ,C4bx, " (pF) "
        print "L5 = " ,L5bx, " (nH) "
        print "G6 = " ,G6b, " (Ohm) "

elif filter_order==6:

        C1ax = 1000000000000/(output_resistance*cutoff_frequency*a1)
        L2ax = (output_resistance*1000000000)/(cutoff_frequency*a2)
        C3ax = 1000000000000/(output_resistance*cutoff_frequency*a3)
        L4ax = (output_resistance*1000000000)/(cutoff_frequency*a4)
        C5ax = 1000000000000/(output_resistance*cutoff_frequency*a5)
        L6ax = (output_resistance*1000000000)/(cutoff_frequency*a6)
        G7a = a7*output_resistance
```

```
        L1bx = (output_resistance*1000000000)/(cutoff_frequer cy*a1)
        C2bx = 1000000000000/(output_resistance*cutoff_frequency*a2)
        L3bx = (output_resistance*1000000000)/(cutoff_frequency*a3)
        C4bx = 1000000000000/(output_resistance*cutoff_frequency*a4)
        L5bx = (output_resistance*1000000000)/(cutoff_frequency*a5)
        C6bx = 1000000000000/(output_resistance*cutoff_frequency*a6)
        G7b = a7*output_resistance

        print "\nType I Filter\n"
        print "C1 = " ,C1ax, " (pF) "
        print "L2 = " ,L2ax, " (nH) "
        print "C3 = " ,C3ax, " (pF) "
        print "L4 = " ,L4ax, " (nH) "
        print "C5 = " ,C5ax, " (pF) "
        print "L6 = " ,L6ax, " (nH) "
        print "G7 = " ,G7a, " (Ohm) "


        print "\nType II Filter\n"
        print "L1 = " ,L1bx, " (nH) "
        print "C2 = " ,C2bx, " (pF) "
        print "L3 = " ,L3bx, " (nH) "
        print "C4 = " ,C4bx, " (pF) "
        print "L5 = " ,L5bx, " (nH) "
        print "C6 = " ,C6bx, " (pF) "
        print "G7 = " ,G7b, " (Ohm) "

elif filter_order==7:

        C1ax = 1000000000000/(output_resistance*cutoff_frequency*a1)
        L2ax = (output_resistance*1000000000)/(cutoff_frequency*a2)
        C3ax = 1000000000000/(output_resistance*cutoff_frequency*a3)
        L4ax = (output_resistance*1000000000)/(cutoff_frequency*a4)
        C5ax = 1000000000000/(output_resistance*cutoff_frequency*a5)
        L6ax = (output_resistance*1000000000)/(cutoff_frequency*a6)
        C7ax = 1000000000000/(output_resistance*cutoff_frequency*a7)
        G8a = a8*output_resistance

        L1bx = (output_resistance*1000000000)/(cutoff_frequency*a1)
        C2bx = 1000000000000/(output_resistance*cutoff_frequency*a2)
        L3bx = (output_resistance*1000000000)/(cutoff_frequency*a3)
        C4bx = 1000000000000/(output_resistance*cutoff_frequency*a4)
        L5bx = (output_resistance*1000000000)/(cutoff_frequency*a5)
        C6bx = 1000000000000/(output_resistance*cutoff_frequency*a6)
        L7bx = (output_resistance*1000000000)/(cutoff_frequency*a7)
        G8b = a8*output_resistance

        print "\nType I Filter\n"
        print "C1 = " ,C1ax, " (pF) "
        print "L2 = " ,L2ax, " (nH) "
        print "C3 = " ,C3ax, " (pF) "
        print "L4 = " ,L4ax, " (nH) "
        print "C5 = " ,C5ax, " (pF) "
        print "L6 = " ,L6ax, " (nH) "
```

```
        print "C7 = " ,C7ax, " (pF) "
        print "G8 = " ,G8a, " (Ohm) "


        print "\nType II Filter\n"
        print "L1 = " ,L1bx, " (nH) "
        print "C2 = " ,C2bx, " (pF) "
        print "L3 = " ,L3bx, " (nH) "
        print "C4 = " ,C4bx, " (pF) "
        print "L5 = " ,L5bx, " (nH) "
        print "C6 = " ,C6bx, " (pF) "
        print "L7 = " ,L7bx, " (nH) "
        print "G8 = " ,G8b, " (Ohm) "

elif filter_order==8:

        C1ax = 1000000000000/(output_resistance*cutoff_frequency*a1)
        L2ax = (output_resistance*1000000000)/(cutoff_frequency*a2)
        C3ax = 1000000000000/(output_resistance*cutoff_frequency*a3)
        L4ax = (output_resistance*1000000000)/(cutoff_frequency*a4)
        C5ax = 1000000000000/(output_resistance*cutoff_frequency*a5)
        L6ax = (output_resistance*1000000000)/(cutoff_frequency*a6)
        C7ax = 1000000000000/(output_resistance*cutoff_frequency*a7)
        L8ax = (output_resistance*1000000000)/(cutoff_frequency*a8)
        G9a = a9*output_resistance

        L1bx = (output_resistance*1000000000)/(cutoff_frequency*a1)
        C2bx = 1000000000000/(output_resistance*cutoff_frequency*a2)
        L3bx = (output_resistance*1000000000)/(cutoff_frequency*a3)
        C4bx = 1000000000000/(output_resistance*cutoff_frequency*a4)
        L5bx = (output_resistance*1000000000)/(cutoff_frequency*a5)
        C6bx = 1000000000000/(output_resistance*cutoff_frequency*a6)
        L7bx = (output_resistance*1000000000)/(cutoff_frequency*a7)
        C8bx = 1000000000000/(output_resistance*cutoff_frequency*a8)
        G9b = a9*output_resistance

        print "\nType I Filter\n"
        print "C1 = " ,C1ax, " (pF) "
        print "L2 = " ,L2ax, " (nH) "
        print "C3 = " ,C3ax, " (pF) "
        print "L4 = " ,L4ax, " (nH) "
        print "C5 = " ,C5ax, " (pF) "
        print "L6 = " ,L6ax, " (nH) "
        print "C7 = " ,C7ax, " (pF) "
        print "L8 = " ,L8ax, " (nH) "
        print "G9 = " ,G9a, " (Ohm) "


        print "\nType II Filter\n"
        print "L1 = " ,L1bx, " (nH) "
        print "C2 = " ,C2bx, " (pF) "
        print "L3 = " ,L3bx, " (nH) "
        print "C4 = " ,C4bx, " (pF) "
        print "L5 = " ,L5bx, " (nH) "
```

```
        print "C6 = " ,C6bx, " (pF) "
        print "L7 = " ,L7bx, " (nH) "
        print "C8 = " ,C8bx, " (pF) "
        print "G9 = " ,G9b, " (Ohm) "

elif filter_order==9:

        C1ax = 1000000000000/(output_resistance*cutoff_frequency*a1)
        L2ax = (output_resistance*1000000000)/(cutoff_frequency*a2)
        C3ax = 1000000000000/(output_resistance*cutoff_frequency*a3)
        L4ax = (output_resistance*1000000000)/(cutoff_frequency*a4)
        C5ax = 1000000000000/(output_resistance*cutoff_frequency*a5)
        L6ax = (output_resistance*1000000000)/(cutoff_frequency*a6)
        C7ax = 1000000000000/(output_resistance*cutoff_frequency*a7)
        L8ax = (output_resistance*1000000000)/(cutoff_frequency*a8)
        C9ax = 1000000000000/(output_resistance*cutoff_frequency*a9)
        G10a = a10*output_resistance

        L1bx = (output_resistance*1000000000)/(cutoff_frequency*a1)
        C2bx = 1000000000000/(output_resistance*cutoff_frequency*a2)
        L3bx = (output_resistance*1000000000)/(cutoff_frequency*a3)
        C4bx = 1000000000000/(output_resistance*cutoff_frequency*a4)
        L5bx = (output_resistance*1000000000)/(cutoff_frequency*a5)
        C6bx = 1000000000000/(output_resistance*cutoff_frequency*a6)
        L7bx = (output_resistance*1000000000)/(cutoff_frequency*a7)
        C8bx = 1000000000000/(output_resistance*cutoff_frequency*a8)
        L9bx = (output_resistance*1000000000)/(cutoff_frequency*a9)
        G10b = a10*output_resistance

        print "\nType I Filter\n"
        print "C1 = " ,C1ax, " (pF) "
        print "L2 = " ,L2ax, " (nH) "
        print "C3 = " ,C3ax, " (pF) "
        print "L4 = " ,L4ax, " (nH) "
        print "C5 = " ,C5ax, " (pF) "
        print "L6 = " ,L6ax, " (nH) "
        print "C7 = " ,C7ax, " (pF) "
        print "L8 = " ,L8ax, " (nH) "
        print "C9 = " ,C9ax, " (pF) "
        print "G10 = " ,G10a, " (Ohm) "


        print "\nType II Filter\n"
        print "L1 = " ,L1bx, " (nH) "
        print "C2 = " ,C2bx, " (pF) "
        print "L3 = " ,L3bx, " (nH) "
        print "C4 = " ,C4bx, " (pF) "
        print "L5 = " ,L5bx, " (nH) "
        print "C6 = " ,C6bx, " (pF) "
        print "L7 = " ,L7bx, " (nH) "
        print "C8 = " ,C8bx, " (pF) "
        print "L9 = " ,L9bx, " (nH) "
        print "G10 = " ,G10b, " (Ohm) "
```

```
elif filter_order==10:

    C1ax = 1000000000000/(output_resistance*cutoff_frequency*a1)
    L2ax = (output_resistance*1000000000)/(cutoff_frequency*a2)
    C3ax = 1000000000000/(output_resistance*cutoff_frequency*a3)
    L4ax = (output_resistance*1000000000)/(cutoff_frequency*a4)
    C5ax = 1000000000000/(output_resistance*cutoff_frequency*a5)
    L6ax = (output_resistance*1000000000)/(cutoff_frequency*a6)
    C7ax = 1000000000000/(output_resistance*cutoff_frequency*a7)
    L8ax = (output_resistance*1000000000)/(cutoff_frequency*a8)
    C9ax = 1000000000000/(output_resistance*cutoff_frequency*a9)
    L10ax = (output_resistance*1000000000)/(cutoff_frequency*a10)
    G11a = a11*output_resistance

    L1bx = (output_resistance*1000000000)/(cutoff_frequency*a1)
    C2bx = 1000000000000/(output_resistance*cutoff_frequency*a2)
    L3bx = (output_resistance*1000000000)/(cutoff_frequency*a3)
    C4bx = 1000000000000/(output_resistance*cutoff_frequency*a4)
    L5bx = (output_resistance*1000000000)/(cutoff_frequency*a5)
    C6bx = 1000000000000/(output_resistance*cutoff_frequency*a6)
    L7bx = (output_resistance*1000000000)/(cutoff_frequency*a7)
    C8bx = 1000000000000/(output_resistance*cutoff_frequency*a8)
    L9bx = (output_resistance*1000000000)/(cutoff_frequency*a9)
    C10bx = 1000000000000/(output_resistance*cutoff_frequency*a10)
    G11b = a11*output_resistance

    print "\nType I Filter\n"
    print "C1 = " ,C1ax, " (pF) "
    print "L2 = " ,L2ax, " (nH) "
    print "C3 = " ,C3ax, " (pF) "
    print "L4 = " ,L4ax, " (nH) "
    print "C5 = " ,C5ax, " (pF) "
    print "L6 = " ,L6ax, " (nH) "
    print "C7 = " ,C7ax, " (pF) "
    print "L8 = " ,L8ax, " (nH) "
    print "C9 = " ,C9ax, " (pF) "
    print "L10 = " ,L10ax, " (nH) "
    print "G11 = " ,G11a, " (Ohm) "


    print "\nType II Filter\n"
    print "L1 = " ,L1bx, " (nH) "
    print "C2 = " ,C2bx, " (pF) "
    print "L3 = " ,L3bx, " (nH) "
    print "C4 = " ,C4bx, " (pF) "
    print "L5 = " ,L5bx, " (nH) "
    print "C6 = " ,C6bx, " (pF) "
    print "L7 = " ,L7bx, " (nH) "
    print "C8 = " ,C8bx, " (pF) "
    print "L9 = " ,L9bx, " (nH) "
    print "C10 = " ,C10bx, " (pF) "
    print "G11 = " ,G11b, " (Ohm) "

else:
```

```
            print "Please enter filter order between 1 to 10"


#######################################################################
#
# Code below will determine which filter to use
# Low pass, highpass, band pass, or band stop filter
# Also assign filter order needed
#
# BAND PASS FILTER
#######################################################################

    elif filter_type ==3:

        frequency_f_lower = input("Enter the Lower Frequency desired(Hz): ")
        frequency_f_upper = input("Enter the Upper Frequency desired(Hz): ")
        frequency_lower = 2*pi*frequency_f_lower
        frequency_upper = 2*pi*frequency_f_upper
        frequency_centre = sqrt(frequency_lower*frequency_upper)
        delta = (frequency_upper-frequency_lower)/frequency_centre

        if filter_order==1:

            L1a = (a1*1000000000)/(frequency_centre*delta) #substitute L1 in low pass filter(Type I)
            C1a = (delta*1000000000000)/(frequency_centre*a1) #substitute L1 in low pass
filter(Type I)
            G2a = a2*output_resistance

            L1b = (delta*1000000000)/(frequency_centre*a1) #substitute C1 in low pass filter(Type
II)
            C1b = (a1*1000000000000)/(frequency_centre*delta) #substitute C1 in low pass
filter(Type II)
            G2b = a2*output_resistance

            print "\nType I Filter\n"
            print "L1 = " ,L1a, " (nH) "
            print "C1 = " ,C1a, " (pF) "
            print "G2 = " ,G2a, " (Ohm) "

            print "\nType II Filter\n"
            print "L1 = " ,L1b, " (nH) "
            print "C1 = " ,C1b, " (pF) "
            print "G2 = " ,G2b, " (Ohm) "

        elif filter_order==2:

            L1a = (a1*1000000000)/(frequency_centre*delta) #substitute L1 in low pass filter(Type I)
            C1a = (delta*1000000000000)/(frequency_centre*a1) #substitute L1 in low pass
filter(Type I)
            L2a = (delta*1000000000)/(frequency_centre*a2) #substitute C2 in low pass filter(Type I)
            C2a = (a2*1000000000000)/(frequency_centre*delta) #substitute C2 in low pass
filter(Type I)
            G3a = a3*output_resistance
```

```
            L1b = (delta*1000000000)/(frequency_centre*a1) #substitute C1 in low pass filter(Type
II)
            C1b = (a1*1000000000000)/(frequency_centre*delta) #substitute C1 in low pass
filter(Type II)
            L2b = (a2*1000000000)/(frequency_centre*delta) #substitute L2 in low pass filter(Type
II)
            C2b = (delta*1000000000000)/(frequency_centre*a2) #substitute L2 in low pass
filter(Type II)
            G3b = a3*output_resistance

            print "\nType I Filter\n"
            print "L1 = " ,L1a, " (nH) "
            print "C1 = " ,C1a, " (pF) "
            print "L2 = " ,L2a, " (nH) "
            print "C2 = " ,C2a, " (pF) "
            print "G3 = " ,G3a, " (Ohm) "

            print "\nType II Filter\n"
            print "L1 = " ,L1b, " (nH) "
            print "C1 = " ,C1b, " (pF) "
            print "L2 = " ,L2b, " (nH) "
            print "C2 = " ,C2b, " (pF) "
            print "G3 = " ,G3b, " (Ohm) "

        elif filter_order==3:

            L1a = (a1*1000000000)/(frequency_centre*delta) #substitute L1 in low pass filter(Type I)
            C1a = (delta*1000000000000)/(frequency_centre*a1) #substitute L1 in low pass
filter(Type I)
            L2a = (delta*1000000000)/(frequency_centre*a2) #substitute C2 in low pass filter(Type I)
            C2a = (a2*1000000000000)/(frequency_centre*delta) #substitute C2 in low pass
filter(Type I)

            L3a = (a3*1000000000)/(frequency_centre*delta)
            C3a = (delta*1000000000000)/(frequency_centre*a3)
            G4a = a4*output_resistance

            L1b = (delta*1000000000)/(frequency_centre*a1) #substitute C1 in low pass filter(Type
II)
            C1b = (a1*1000000000000)/(frequency_centre*delta) #substitute C1 in low pass
filter(Type II)
            L2b = (a2*1000000000)/(frequency_centre*delta) #substitute L2 in low pass filter(Type
II)
            C2b = (delta*1000000000000)/(frequency_centre*a2) #substitute L2 in low pass
filter(Type II)

            L3b = (delta*1000000000)/(frequency_centre*a3)
            C3b = (a3*1000000000000)/(frequency_centre*delta)
            G4b = a4*output_resistance

            print "\nType I Filter\n"
            print "L1 = " ,L1a, " (nH) "
            print "C1 = " ,C1a, " (pF) "
            print "L2 = " ,L2a, " (nH) "
```

```
            print "C2 = " ,C2a, " (pF) "

            print "L3 = " ,L3a, " (nH) "
            print "C3 = " ,C3a, " (pF) "
            print "G4 = " ,G4a, " (Ohm) "

            print "\nType II Filter\n"
            print "L1 = " ,L1b, " (nH) "
            print "C1 = " ,C1b, " (pF) "
            print "L2 = " ,L2b, " (nH) "
            print "C2 = " ,C2b, " (pF) "

            print "L3 = " ,L3b, " (nH) "
            print "C3 = " ,C3b, " (pF) "
            print "G4 = " ,G4b, " (Ohm) "

        elif filter_order==4:

            L1a = (a1*1000000000)/(frequency_centre*delta) #substitute L1 in low pass filter(Type I)
            C1a = (delta*1000000000000)/(frequency_centre*a1) #substitute L1 in low pass
filter(Type I)
            L2a = (delta*1000000000)/(frequency_centre*a2) #substitute C2 in low pass filter(Type I)
            C2a = (a2*1000000000000)/(frequency_centre*delta) #substitute C2 in low pass
filter(Type I)

            L3a = (a3*1000000000)/(frequency_centre*delta)
            C3a = (delta*1000000000000)/(frequency_centre*a3)
            L4a = (delta*1000000000)/(frequency_centre*a4)
            C4a = (a4*1000000000000)/(frequency_centre*delta)
            G5a = a11*output_resistance

            L1b = (delta*1000000000)/(frequency_centre*a1) #substitute C1 in low pass filter(Type
II)
            C1b = (a1*1000000000000)/(frequency_centre*delta) #substitute C1 in low pass
filter(Type II)
            L2b = (a2*1000000000)/(frequency_centre*delta) #substitute L2 in low pass filter(Type
II)
            C2b = (delta*1000000000000)/(frequency_centre*a2) #substitute L2 in low pass
filter(Type II)

            L3b = (delta*1000000000)/(frequency_centre*a3)
            C3b = (a3*1000000000000)/(frequency_centre*delta)
            L4b = (a4*1000000000)/(frequency_centre*delta)
            C4b = (delta*1000000000000)/(frequency_centre*a4)
            G5b = a11*output_resistance

            print "\nType I Filter\n"
            print "L1 = " ,L1a, " (nH) "
            print "C1 = " ,C1a, " (pF) "
            print "L2 = " ,L2a, " (nH) "
            print "C2 = " ,C2a, " (pF) "

            print "L3 = " ,L3a, " (nH) "
            print "C3 = " ,C3a, " (pF) "
```

```
            print "L4 = " ,L4a, " (nH) "
            print "C4 = " ,C4a, " (pF) "
            print "G5 = " ,G5a, " (Ohm) "

            print "\nType II Filter\n"
            print "L1 = " ,L1b, " (nH) "
            print "C1 = " ,C1b, " (pF) "
            print "L2 = " ,L2b, " (nH) "
            print "C2 = " ,C2b, " (pF) "

            print "L3 = " ,L3b, " (nH) "
            print "C3 = " ,C3b, " (pF) "
            print "L4 = " ,L4b, " (nH) "
            print "C4 = " ,C4b, " (pF) "
            print "G5 = " ,G5b, " (Ohm) "

        elif filter_order==5:

            L1a = (a1*1000000000)/(frequency_centre*delta) #substitute L1 in low pass filter(Type I)
            C1a = (delta*1000000000000)/(frequency_centre*a1) #substitute L1 in low pass
filter(Type I)
            L2a = (delta*1000000000)/(frequency_centre*a2) #substitute C2 in low pass filter(Type I)
            C2a = (a2*1000000000000)/(frequency_centre*delta) #substitute C2 in low pass
filter(Type I)

            L3a = (a3*1000000000)/(frequency_centre*delta)
            C3a = (delta*1000000000000)/(frequency_centre*a3)
            L4a = (delta*1000000000)/(frequency_centre*a4)
            C4a = (a4*1000000000000)/(frequency_centre*delta)

            L5a = (a5*1000000000)/(frequency_centre*delta)
            C5a = (delta*1000000000000)/(frequency_centre*a5)
            G6a = a6*output_resistance

            L1b = (delta*1000000000)/(frequency_centre*a1) #substitute C1 in low pass filter(Type
II)
            C1b = (a1*1000000000000)/(frequency_centre*delta) #substitute C1 in low pass
filter(Type II)
            L2b = (a2*1000000000)/(frequency_centre*delta) #substitute L2 in low pass filter(Type
II)
            C2b = (delta*1000000000000)/(frequency_centre*a2) #substitute L2 in low pass
filter(Type II)

            L3b = (delta*1000000000)/(frequency_centre*a3)
            C3b = (a3*1000000000000)/(frequency_centre*delta)
            L4b = (a4*1000000000)/(frequency_centre*delta)
            C4b = (delta*1000000000000)/(frequency_centre*a4)

            L5b = (delta*1000000000)/(frequency_centre*a5)
            C5b = (a5*1000000000000)/(frequency_centre*delta)
            G6b = a6*output_resistance

            print "\nType I Filter\n"
            print "L1 = " ,L1a, " (nH) "
```

```
            print "C1 = " ,C1a, " (pF) "
            print "L2 = " ,L2a, " (nH) "
            print "C2 = " ,C2a, " (pF) "

            print "L3 = " ,L3a, " (nH) "
            print "C3 = " ,C3a, " (pF) "
            print "L4 = " ,L4a, " (nH) "
            print "C4 = " ,C4a, " (pF) "

            print "L5 = " ,L5a, " (nH) "
            print "C5 = " ,C5a, " (pF) "
            print "G6 = " ,G6a, " (Ohm) "

            print "\nType II Filter\n"
            print "L1 = " ,L1b, " (nH) "
            print "C1 = " ,C1b, " (pF) "
            print "L2 = " ,L2b, " (nH) "
            print "C2 = " ,C2b, " (pF) "

            print "L3 = " ,L3b, " (nH) "
            print "C3 = " ,C3b, " (pF) "
            print "L4 = " ,L4b, " (nH) "
            print "C4 = " ,C4b, " (pF) "

            print "L5 = " ,L5b, " (nH) "
            print "C5 = " ,C5b, " (pF) "
            print "G6 = " ,G6b, " (Ohm) "

        elif filter_order==6:

            L1a = (a1*1000000000)/(frequency_centre*delta) #substitute L1 in low pass filter(Type I)
            C1a = (delta*1000000000000)/(frequency_centre*a1) #substitute L1 in low pass
filter(Type I)
            L2a = (delta*1000000000)/(frequency_centre*a2) #substitute C2 in low pass filter(Type I)
            C2a = (a2*1000000000000)/(frequency_centre*delta) #substitute C2 in low pass
filter(Type I)

            L3a = (a3*1000000000)/(frequency_centre*delta)
            C3a = (delta*1000000000000)/(frequency_centre*a3)
            L4a = (delta*1000000000)/(frequency_centre*a4)
            C4a = (a4*1000000000000)/(frequency_centre*delta)

            L5a = (a5*1000000000)/(frequency_centre*delta)
            C5a = (delta*1000000000000)/(frequency_centre*a5)
            L6a = (delta*1000000000)/(frequency_centre*a6)
            C6a = (a6*1000000000000)/(frequency_centre*delta)
            G7a = a7*output_resistance

            L1b = (delta*1000000000)/(frequency_centre*a1) #substitute C1 in low pass filter(Type
II)
            C1b = (a1*1000000000000)/(frequency_centre*delta) #substitute C1 in low pass
filter(Type II)
            L2b = (a2*1000000000)/(frequency_centre*delta) #substitute L2 in low pass filter(Type
II)
```

```
        C2b = (delta*1000000000000)/(frequency_centre*a2) #substitute L2 in low pass
filter(Type II)

        L3b = (delta*1000000000)/(frequency_centre*a3)
        C3b = (a3*1000000000000)/(frequency_centre*delta)
        L4b = (a4*1000000000)/(frequency_centre*delta)
        C4b = (delta*1000000000000)/(frequency_centre*a4)

        L5b = (delta*1000000000)/(frequency_centre*a5)
        C5b = (a5*1000000000000)/(frequency_centre*delta)
        L6b = (a6*1000000000)/(frequency_centre*delta)
        C6b = (delta*1000000000000)/(frequency_centre*a6)
        G7b = a7*output_resistance

        print "\nType I Filter\n"
        print "L1 = " ,L1a, " (nH) "
        print "C1 = " ,C1a, " (pF) "
        print "L2 = " ,L2a, " (nH) "
        print "C2 = " ,C2a, " (pF) "

        print "L3 = " ,L3a, " (nH) "
        print "C3 = " ,C3a, " (pF) "
        print "L4 = " ,L4a, " (nH) "
        print "C4 = " ,C4a, " (pF) "

        print "L5 = " ,L5a, " (nH) "
        print "C5 = " ,C5a, " (pF) "
        print "L6 = " ,L6a, " (nH) "
        print "C6 = " ,C6a, " (pF) "
        print "G7 = " ,G7a, " (Ohm) "

        print "\nType II Filter\n"
        print "L1 = " ,L1b, " (nH) "
        print "C1 = " ,C1b, " (pF) "
        print "L2 = " ,L2b, " (nH) "
        print "C2 = " ,C2b, " (pF) "

        print "L3 = " ,L3b, " (nH) "
        print "C3 = " ,C3b, " (pF) "
        print "L4 = " ,L4b, " (nH) "
        print "C4 = " ,C4b, " (pF) "

        print "L5 = " ,L5b, " (nH) "
        print "C5 = " ,C5b, " (pF) "
        print "L6 = " ,L6b, " (nH) "
        print "C6 = " ,C6b, " (pF) "
        print "G7 = " ,G7b, " (Ohm) "

    elif filter_order==7:

        L1a = (a1*1000000000)/(frequency_centre*delta) #substitute L1 in low pass filter(Type I)
        C1a = (delta*1000000000000)/(frequency_centre*a1) #substitute L1 in low pass
filter(Type I)
        L2a = (delta*1000000000)/(frequency_centre*a2) #substitute C2 in low pass filter(Type I)
```

```
        C2a = (a2*1000000000000)/(frequency_centre*delta) #substitute C2 in low pass
filter(Type I)

        L3a = (a3*1000000000)/(frequency_centre*delta)
        C3a = (delta*1000000000000)/(frequency_centre*a3)
        L4a = (delta*1000000000)/(frequency_centre*a4)
        C4a = (a4*1000000000000)/(frequency_centre*delta)

        L5a = (a5*1000000000)/(frequency_centre*delta)
        C5a = (delta*1000000000000)/(frequency_centre*a5)
        L6a = (delta*1000000000)/(frequency_centre*a6)
        C6a = (a6*1000000000000)/(frequency_centre*delta)

        L7a = (a7*1000000000)/(frequency_centre*delta)
        C7a = (delta*1000000000000)/(frequency_centre*a7)
        G8a = a8*output_resistance

        L1b = (delta*1000000000)/(frequency_centre*a1) #substitute C1 in low pass filter(Type
II)
        C1b = (a1*1000000000000)/(frequency_centre*delta) #substitute C1 in low pass
filter(Type II)
        L2b = (a2*1000000000)/(frequency_centre*delta) #substitute L2 in low pass filter(Type
II)
        C2b = (delta*1000000000000)/(frequency_centre*a2) #substitute L2 in low pass
filter(Type II)

        L3b = (delta*1000000000)/(frequency_centre*a3)
        C3b = (a3*1000000000000)/(frequency_centre*delta)
        L4b = (a4*1000000000)/(frequency_centre*delta)
        C4b = (delta*1000000000000)/(frequency_centre*a4)

        L5b = (delta*1000000000)/(frequency_centre*a5)
        C5b = (a5*1000000000000)/(frequency_centre*delta)
        L6b = (a6*1000000000)/(frequency_centre*delta)
        C6b = (delta*1000000000000)/(frequency_centre*a6)

        L7b = (delta*1000000000)/(frequency_centre*a7)
        C7b = (a7*1000000000000)/(frequency_centre*delta)
        G8b = a8*output_resistance

        print "\nType I Filter\n"
        print "L1 = " ,L1a, " (nH) "
        print "C1 = " ,C1a, " (pF) "
        print "L2 = " ,L2a, " (nH) "
        print "C2 = " ,C2a, " (pF) "

        print "L3 = " ,L3a, " (nH) "
        print "C3 = " ,C3a, " (pF) "
        print "L4 = " ,L4a, " (nH) "
        print "C4 = " ,C4a, " (pF) "

        print "L5 = " ,L5a, " (nH) "
        print "C5 = " ,C5a, " (pF) "
        print "L6 = " ,L6a, " (nH) "
```

```
        print "C6 = " ,C6a, " (pF) "

        print "L7 = " ,L7a, " (nH) "
        print "C7 = " ,C7a, " (pF) "
        print "G8 = " ,G8a, " (Ohm) "

        print "\nType II Filter\n"
        print "L1 = " ,L1b, " (nH) "
        print "C1 = " ,C1b, " (pF) "
        print "L2 = " ,L2b, " (nH) "
        print "C2 = " ,C2b, " (pF) "

        print "L3 = " ,L3b, " (nH) "
        print "C3 = " ,C3b, " (pF) "
        print "L4 = " ,L4b, " (nH) "
        print "C4 = " ,C4b, " (pF) "

        print "L5 = " ,L5b, " (nH) "
        print "C5 = " ,C5b, " (pF) "
        print "L6 = " ,L6b, " (nH) "
        print "C6 = " ,C6b, " (pF) "

        print "L7 = " ,L7b, " (nH) "
        print "C7 = " ,C7b, " (pF) "
        print "G8 = " ,G8b, " (Ohm) "

    elif filter_order==8:

        L1a = (a1*1000000000)/(frequency_centre*delta) #substitute L1 in low pass filter(Type I)
        C1a = (delta*1000000000000)/(frequency_centre*a1) #substitute L1 in low pass
filter(Type I)
        L2a = (delta*1000000000)/(frequency_centre*a2) #substitute C2 in low pass filter(Type I)
        C2a = (a2*1000000000000)/(frequency_centre*delta) #substitute C2 in low pass
filter(Type I)

        L3a = (a3*1000000000)/(frequency_centre*delta)
        C3a = (delta*1000000000000)/(frequency_centre*a3)
        L4a = (delta*1000000000)/(frequency_centre*a4)
        C4a = (a4*1000000000000)/(frequency_centre*delta)

        L5a = (a5*1000000000)/(frequency_centre*delta)
        C5a = (delta*1000000000000)/(frequency_centre*a5)
        L6a = (delta*1000000000)/(frequency_centre*a6)
        C6a = (a6*1000000000000)/(frequency_centre*delta)

        L7a = (a7*1000000000)/(frequency_centre*delta)
        C7a = (delta*1000000000000)/(frequency_centre*a7)
        L8a = (delta*1000000000)/(frequency_centre*a8)
        C8a = (a8*1000000000000)/(frequency_centre*delta)
        G9a = a9*output_resistance

        L1b = (delta*1000000000)/(frequency_centre*a1) #substitute C1 in low pass filter(Type
II)
```

```
C1b = (a1*1000000000000)/(frequency_centre*delta) #substitute C1 in low pass
filter(Type II)
L2b = (a2*1000000000)/(frequency_centre*delta) #substitute L2 in low pass filter(Type
II)
C2b = (delta*1000000000000)/(frequency_centre*a2) #substitute L2 in low pass
filter(Type II)

L3b = (delta*1000000000)/(frequency_centre*a3)
C3b = (a3*1000000000000)/(frequency_centre*delta)
L4b = (a4*1000000000)/(frequency_centre*delta)
C4b = (delta*1000000000000)/(frequency_centre*a4)

L5b = (delta*1000000000)/(frequency_centre*a5)
C5b = (a5*1000000000000)/(frequency_centre*delta)
L6b = (a6*1000000000)/(frequency_centre*delta)
C6b = (delta*1000000000000)/(frequency_centre*a6)

L7b = (delta*1000000000)/(frequency_centre*a7)
C7b = (a7*1000000000000)/(frequency_centre*delta)
L8b = (a8*1000000000)/(frequency_centre*delta)
C8b = (delta*1000000000000)/(frequency_centre*a8)
G9b = a9*output_resistance

print "\nType I Filter\n"
print "L1 = " ,L1a, " (nH) "
print "C1 = " ,C1a, " (pF) "
print "L2 = " ,L2a, " (nH) "
print "C2 = " ,C2a, " (pF) "

print "L3 = " ,L3a, " (nH) "
print "C3 = " ,C3a, " (pF) "
print "L4 = " ,L4a, " (nH) "
print "C4 = " ,C4a, " (pF) "

print "L5 = " ,L5a, " (nH) "
print "C5 = " ,C5a, " (pF) "
print "L6 = " ,L6a, " (nH) "
print "C6 = " ,C6a, " (pF) "

print "L7 = " ,L7a, " (nH) "
print "C7 = " ,C7a, " (pF) "
print "L8 = " ,L8a, " (nH) "
print "C8 = " ,C8a, " (pF) "
print "G9 = " ,G9a, " (Ohm) "

print "\nType II Filter\n"
print "L1 = " ,L1b, " (nH) "
print "C1 = " ,C1b, " (pF) "
print "L2 = " ,L2b, " (nH) "
print "C2 = " ,C2b, " (pF) "

print "L3 = " ,L3b, " (nH) "
print "C3 = " ,C3b, " (pF) "
print "L4 = " ,L4b, " (nH) "
```

```
            print "C4 = " ,C4b, " (pF) "

            print "L5 = " ,L5b, " (nH) "
            print "C5 = " ,C5b, " (pF) "
            print "L6 = " ,L6b, " (nH) "
            print "C6 = " ,C6b, " (pF) "

            print "L7 = " ,L7b, " (nH) "
            print "C7 = " ,C7b, " (pF) "
            print "L8 = " ,L8b, " (nH) "
            print "C8 = " ,C8b, " (pF) "
            print "G9 = " ,G9b, " (Ohm) "

        elif filter_order==9:

            L1a = (a1*1000000000)/(frequency_centre*delta) #substitute L1 in low pass filter(Type I)
            C1a = (delta*1000000000000)/(frequency_centre*a1) #substitute L1 in low pass
filter(Type I)
            L2a = (delta*1000000000)/(frequency_centre*a2) #substitute C2 in low pass filter(Type I)
            C2a = (a2*1000000000000)/(frequency_centre*delta) #substitute C2 in low pass
filter(Type I)

            L3a = (a3*1000000000)/(frequency_centre*delta)
            C3a = (delta*1000000000000)/(frequency_centre*a3)
            L4a = (delta*1000000000)/(frequency_centre*a4)
            C4a = (a4*1000000000000)/(frequency_centre*delta)

            L5a = (a5*1000000000)/(frequency_centre*delta)
            C5a = (delta*1000000000000)/(frequency_centre*a5)
            L6a = (delta*1000000000)/(frequency_centre*a6)
            C6a = (a6*1000000000000)/(frequency_centre*delta)

            L7a = (a7*1000000000)/(frequency_centre*delta)
            C7a = (delta*1000000000000)/(frequency_centre*a7)
            L8a = (delta*1000000000)/(frequency_centre*a8)
            C8a = (a8*1000000000000)/(frequency_centre*delta)

            L9a = (a9*1000000000)/(frequency_centre*delta)
            C9a = (delta*1000000000000)/(frequency_centre*a9)
            G10a = a10*output_resistance

            L1b = (delta*1000000000)/(frequency_centre*a1) #substitute C1 in low pass filter(Type
II)
            C1b = (a1*1000000000000)/(frequency_centre*delta) #substitute C1 in low pass
filter(Type II)
            L2b = (a2*1000000000)/(frequency_centre*delta) #substitute L2 in low pass filter(Type
II)
            C2b = (delta*1000000000000)/(frequency_centre*a2) #substitute L2 in low pass
filter(Type II)

            L3b = (delta*1000000000)/(frequency_centre*a3)
            C3b = (a3*1000000000000)/(frequency_centre*delta)
            L4b = (a4*1000000000)/(frequency_centre*delta)
            C4b = (delta*1000000000000)/(frequency_centre*a4)
```

```
L5b = (delta*1000000000)/(frequency_centre*a5)
C5b = (a5*1000000000000)/(frequency_centre*delta)
L6b = (a6*1000000000)/(frequency_centre*delta)
C6b = (delta*1000000000000)/(frequency_centre*a6)

L7b = (delta*1000000000)/(frequency_centre*a7)
C7b = (a7*1000000000000)/(frequency_centre*delta)
L8b = (a8*1000000000)/(frequency_centre*delta)
C8b = (delta*1000000000000)/(frequency_centre*a8)

L9b = (delta*1000000000)/(frequency_centre*a9)
C9b = (a9*1000000000000)/(frequency_centre*delta)
G10b = a10*output_resistance

print "\nType I Filter\n"
print "L1 = " ,L1a, " (nH) "
print "C1 = " ,C1a, " (pF) "
print "L2 = " ,L2a, " (nH) "
print "C2 = " ,C2a, " (pF) "

print "L3 = " ,L3a, " (nH) "
print "C3 = " ,C3a, " (pF) "
print "L4 = " ,L4a, " (nH) "
print "C4 = " ,C4a, " (pF) "

print "L5 = " ,L5a, " (nH) "
print "C5 = " ,C5a, " (pF) "
print "L6 = " ,L6a, " (nH) "
print "C6 = " ,C6a, " (pF) "

print "L7 = " ,L7a, " (nH) "
print "C7 = " ,C7a, " (pF) "
print "L8 = " ,L8a, " (nH) "
print "C8 = " ,C8a, " (pF) "

print "L9 = " ,L9a, " (nH) "
print "C9 = " ,C9a, " (pF) "
print "G10 = " ,G10a, " (Ohm) "

print "\nType II Filter\n"
print "L1 = " ,L1b, " (nH) "
print "C1 = " ,C1b, " (pF) "
print "L2 = " ,L2b, " (nH) "
print "C2 = " ,C2b, " (pF) "

print "L3 = " ,L3b, " (nH) "
print "C3 = " ,C3b, " (pF) "
print "L4 = " ,L4b, " (nH) "
print "C4 = " ,C4b, " (pF) "

print "L5 = " ,L5b, " (nH) "
print "C5 = " ,C5b, " (pF) "
print "L6 = " ,L6b, " (nH) "
```

```
            print "C6 = " ,C6b, " (pF) "

            print "L7 = " ,L7b, " (nH) "
            print "C7 = " ,C7b, " (pF) "
            print "L8 = " ,L8b, " (nH) "
            print "C8 = " ,C8b, " (pF) "

            print "L9 = " ,L9b, " (nH) "
            print "C9 = " ,C9b, " (pF) "
            print "G10 = " ,G10b, " (Ohm) "

        elif filter_order==10:

            L1a = (a1*1000000000)/(frequency_centre*delta) #substitute L1 in low pass filter(Type I)
            C1a = (delta*1000000000000)/(frequency_centre*a1) #substitute L1 in low pass
filter(Type I)
            L2a = (delta*1000000000)/(frequency_centre*a2) #substitute C2 in low pass filter(Type I)
            C2a = (a2*1000000000000)/(frequency_centre*delta) #substitute C2 in low pass
filter(Type I)

            L3a = (a3*1000000000)/(frequency_centre*delta)
            C3a = (delta*1000000000000)/(frequency_centre*a3)
            L4a = (delta*1000000000)/(frequency_centre*a4)
            C4a = (a4*1000000000000)/(frequency_centre*delta)

            L5a = (a5*1000000000)/(frequency_centre*delta)
            C5a = (delta*1000000000000)/(frequency_centre*a5)
            L6a = (delta*1000000000)/(frequency_centre*a6)
            C6a = (a6*1000000000000)/(frequency_centre*delta)

            L7a = (a7*1000000000)/(frequency_centre*delta)
            C7a = (delta*1000000000000)/(frequency_centre*a7)
            L8a = (delta*1000000000)/(frequency_centre*a8)
            C8a = (a8*1000000000000)/(frequency_centre*delta)

            L9a = (a9*1000000000)/(frequency_centre*delta)
            C9a = (delta*1000000000000)/(frequency_centre*a9)
            L10a = (delta*1000000000)/(frequency_centre*a10)
            C10a = (a10*1000000000000)/(frequency_centre*delta)
            G11a = a11*output_resistance

            L1b = (delta*1000000000)/(frequency_centre*a1) #substitute C1 in low pass filter(Type
II)
            C1b = (a1*1000000000000)/(frequency_centre*delta) #substitute C1 in low pass
filter(Type II)
            L2b = (a2*1000000000)/(frequency_centre*delta) #substitute L2 in low pass filter(Type
II)
            C2b = (delta*1000000000000)/(frequency_centre*a2) #substitute L2 in low pass
filter(Type II)

            L3b = (delta*1000000000)/(frequency_centre*a3)
            C3b = (a3*1000000000000)/(frequency_centre*delta)
            L4b = (a4*1000000000)/(frequency_centre*delta)
            C4b = (delta*1000000000000)/(frequency_centre*a4)
```

```
L5b = (delta*1000000000)/(frequency_centre*a5)
C5b = (a5*1000000000000)/(frequency_centre*delta)
L6b = (a6*1000000000)/(frequency_centre*delta)
C6b = (delta*1000000000000)/(frequency_centre*a6)

L7b = (delta*1000000000)/(frequency_centre*a7)
C7b = (a7*1000000000000)/(frequency_centre*delta)
L8b = (a8*1000000000)/(frequency_centre*delta)
C8b = (delta*1000000000000)/(frequency_centre*a8)

L9b = (delta*1000000000)/(frequency_centre*a9)
C9b = (a9*1000000000000)/(frequency_centre*delta)
L10b = (a10*1000000000)/(frequency_centre*delta)
C10b = (delta*1000000000000)/(frequency_centre*a10)
G11b = a11*output_resistance

print "\nType I Filter\n"
print "L1 = " ,L1a, " (nH) "
print "C1 = " ,C1a, " (pF) "
print "L2 = " ,L2a, " (nH) "
print "C2 = " ,C2a, " (pF) "

print "L3 = " ,L3a, " (nH) "
print "C3 = " ,C3a, " (pF) "
print "L4 = " ,L4a, " (nH) "
print "C4 = " ,C4a, " (pF) "

print "L5 = " ,L5a, " (nH) "
print "C5 = " ,C5a, " (pF) "
print "L6 = " ,L6a, " (nH) "
print "C6 = " ,C6a, " (pF) "

print "L7 = " ,L7a, " (nH) "
print "C7 = " ,C7a, " (pF) "
print "L8 = " ,L8a, " (nH) "
print "C8 = " ,C8a, " (pF) "

print "L9 = " ,L9a, " (nH) "
print "C9 = " ,C9a, " (pF) "
print "L10 = " ,L10a, " (nH) "
print "C10 = " ,C10a, " (pF) "
print "G11 = " ,G11a, " (Ohm) "

print "\nType II Filter\n"
print "L1 = " ,L1b, " (nH) "
print "C1 = " ,C1b, " (pF) "
print "L2 = " ,L2b, " (nH) "
print "C2 = " ,C2b, " (pF) "

print "L3 = " ,L3b, " (nH) "
print "C3 = " ,C3b, " (pF) "
print "L4 = " ,L4b, " (nH) "
print "C4 = " ,C4b, " (pF) "
```

```
            print "L5 = " ,L5b, " (nH) "
            print "C5 = " ,C5b, " (pF) "
            print "L6 = " ,L6b, " (nH) "
            print "C6 = " ,C6b, " (pF) "

            print "L7 = " ,L7b, " (nH) "
            print "C7 = " ,C7b, " (pF) "
            print "L8 = " ,L8b, " (nH) "
            print "C8 = " ,C8b, " (pF) "

            print "L9 = " ,L9b, " (nH) "
            print "C9 = " ,C9b, " (pF) "
            print "L10 = " ,L10b, " (nH) "
            print "C10 = " ,C10b, " (pF) "
            print "G11 = " ,G11b, " (Ohm) "

        else:
            print "Please enter filter order between 1 to 10"


########################################################################
#
# Code below will determine which filter to use
# Low pass, highpass, band pass, or band stop filter
# Also assign filter order needed
#
# BAND STOP FILTER
########################################################################

    elif filter_type ==4:

        frequency_f_lower = input("Enter the Lower Frequency desired(Hz): ")
        frequency_f_upper = input("Enter the Upper Frequency desired(Hz): ")
        frequency_lower = 2*pi*frequency_f_lower
        frequency_upper = 2*pi*frequency_f_upper
        frequency_centre = sqrt(frequency_lower*frequency_upper)
        delta = (frequency_upper-frequency_lower)/frequency_centre

        if filter_order==1:

            L1a = (a1*delta*1000000000)/(frequency_centre) #substitute L1 in low pass filter(Type I)
            C1a = (1000000000000)/(delta*frequency_centre*a1) #substitute L1 in low pass
filter(Type I)
            G2a = a2*output_resistance

            L1b = (1000000000)/(delta*frequency_centre*a1) #substitute C1 in low pass filter(Type
II)
            C1b = (a1*delta*1000000000000)/(frequency_centre) #substitute C1 in low pass
filter(Type II)
            G2b = a2*output_resistance

            print "\nType I Filter\n"
            print "L1 = " ,L1a, " (nH) "
```

```
            print "C1 = " ,C1a, " (pF) "
            print "G2 = " ,G2a, " (Ohm) "

            print "\nType II Filter\n"
            print "L1 = " ,L1b, " (nH) "
            print "C1 = " ,C1b, " (pF) "
            print "G2 = " ,G2b, " (Ohm) "

        elif filter_order==2:

            L1a = (a1*delta*1000000000)/(frequency_centre) #substitute L1 in low pass filter(Type I)
            C1a = (1000000000000)/(delta*frequency_centre*a1) #substitute L1 in low pass
filter(Type I)
            L2a = (1000000000)/(delta*frequency_centre*a2) #substitute C2 in low pass filter(Type I)
            C2a = (a2*delta*1000000000000)/(frequency_centre) #substitute C2 in low pass
filter(Type I)
            G3a = a3*output_resistance

            L1b = (1000000000)/(delta*frequency_centre*a1) #substitute C1 in low pass filter(Type
II)
            C1b = (a1*delta*1000000000000)/(frequency_centre) #substitute C1 in low pass
filter(Type II)
            L2b = (a2*delta*1000000000)/(frequency_centre) #substitute L2 in low pass filter(Type
II)
            C2b = (1000000000000)/(delta*frequency_centre*a2) #substitute L2 in low pass
filter(Type II)
            G3b = a3*output_resistance

            print "\nType I Filter\n"
            print "L1 = " ,L1a, " (nH) "
            print "C1 = " ,C1a, " (pF) "
            print "L2 = " ,L2a, " (nH) "
            print "C2 = " ,C2a, " (pF) "
            print "G3 = " ,G3a, " (Ohm) "

            print "\nType II Filter\n"
            print "L1 = " ,L1b, " (nH) "
            print "C1 = " ,C1b, " (pF) "
            print "L2 = " ,L2b, " (nH) "
            print "C2 = " ,C2b, " (pF) "
            print "G3 = " ,G3b, " (Ohm) "

        elif filter_order==3:

            L1a = (a1*delta*1000000000)/(frequency_centre) #substitute L1 in low pass filter(Type I)
            C1a = (1000000000000)/(delta*frequency_centre*a1) #substitute L1 in low pass
filter(Type I)
            L2a = (1000000000)/(delta*frequency_centre*a2) #substitute C2 in low pass filter(Type I)
            C2a = (a2*delta*1000000000000)/(frequency_centre) #substitute C2 in low pass
filter(Type I)

            L3a = (a3*delta*1000000000)/(frequency_centre)
            C3a = (1000000000000)/(delta*frequency_centre*a3)
            G4a = a4*output_resistance
```

```
            L1b = (1000000000)/(delta*frequency_centre*a1) #substitute C1 in low pass filter(Type
II)
            C1b = (a1*delta*1000000000000)/(frequency_centre) #substitute C1 in low pass
filter(Type II)
            L2b = (a2*delta*1000000000)/(frequency_centre) #substitute L2 in low pass filter(Type
II)
            C2b = (1000000000000)/(delta*frequency_centre*a2) #substitute L2 in low pass
filter(Type II)

            L3b = (1000000000)/(delta*frequency_centre*a3)
            C3b = (a3*delta*1000000000000)/(frequency_centre)
            G4b = a4*output_resistance

            print "\nType I Filter\n"
            print "L1 = " ,L1a, " (nH) "
            print "C1 = " ,C1a, " (pF) "
            print "L2 = " ,L2a, " (nH) "
            print "C2 = " ,C2a, " (pF) "

            print "L3 = " ,L3a, " (nH) "
            print "C3 = " ,C3a, " (pF) "
            print "G4 = " ,G4a, " (Ohm) "

            print "\nType II Filter\n"
            print "L1 = " ,L1b, " (nH) "
            print "C1 = " ,C1b, " (pF) "
            print "L2 = " ,L2b, " (nH) "
            print "C2 = " ,C2b, " (pF) "

            print "L3 = " ,L3b, " (nH) "
            print "C3 = " ,C3b, " (pF) "
            print "G4 = " ,G4b, " (Ohm) "

        elif filter_order==4:

            L1a = (a1*delta*1000000000)/(frequency_centre) #substitute L1 in low pass filter(Type I)
            C1a = (1000000000000)/(delta*frequency_centre*a1) #substitute L1 in low pass
filter(Type I)
            L2a = (1000000000)/(delta*frequency_centre*a2) #substitute C2 in low pass filter(Type I)
            C2a = (a2*delta*1000000000000)/(frequency_centre) #substitute C2 in low pass
filter(Type I)

            L3a = (a3*delta*1000000000)/(frequency_centre)
            C3a = (1000000000000)/(delta*frequency_centre*a3)
            L4a = (1000000000)/(delta*frequency_centre*a4)
            C4a = (a4*delta*1000000000000)/(frequency_centre)
            G5a = a5*output_resistance

            L1b = (1000000000)/(delta*frequency_centre*a1) #substitute C1 in low pass filter(Type
II)
            C1b = (a1*delta*1000000000000)/(frequency_centre) #substitute C1 in low pass
filter(Type II)
```

```
            L2b = (a2*delta*1000000000)/(frequency_centre) #substitute L2 in low pass filter(Type
II)
            C2b = (1000000000000)/(delta*frequency_centre*a2) #substitute L2 in low pass
filter(Type II)

            L3b = (1000000000)/(delta*frequency_centre*a3)
            C3b = (a3*delta*1000000000000)/(frequency_centre)
            L4b = (a4*delta*1000000000)/(frequency_centre)
            C4b = (1000000000000)/(delta*frequency_centre*a4)
            G5b = a5*output_resistance

            print "\nType I Filter\n"
            print "L1 = " ,L1a, " (nH) "
            print "C1 = " ,C1a, " (pF) "
            print "L2 = " ,L2a, " (nH) "
            print "C2 = " ,C2a, " (pF) "

            print "L3 = " ,L3a, " (nH) "
            print "C3 = " ,C3a, " (pF) "
            print "L4 = " ,L4a, " (nH) "
            print "C4 = " ,C4a, " (pF) "
            print "G5 = " ,G5a, " (Ohm) "

            print "\nType II Filter\n"
            print "L1 = " ,L1b, " (nH) "
            print "C1 = " ,C1b, " (pF) "
            print "L2 = " ,L2b, " (nH) "
            print "C2 = " ,C2b, " (pF) "

            print "L3 = " ,L3b, " (nH) "
            print "C3 = " ,C3b, " (pF) "
            print "L4 = " ,L4b, " (nH) "
            print "C4 = " ,C4b, " (pF) "
            print "G5 = " ,G5b, " (Ohm) "

        elif filter_order==5:

            L1a = (a1*delta*1000000000)/(frequency_centre) #substitute L1 in low pass filter(Type I)
            C1a = (1000000000000)/(delta*frequency_centre*a1) #substitute L1 in low pass
filter(Type I)
            L2a = (1000000000)/(delta*frequency_centre*a2) #substitute C2 in low pass filter(Type I)
            C2a = (a2*delta*1000000000000)/(frequency_centre) #substitute C2 in low pass
filter(Type I)

            L3a = (a3*delta*1000000000)/(frequency_centre)
            C3a = (1000000000000)/(delta*frequency_centre*a3)
            L4a = (1000000000)/(delta*frequency_centre*a4)
            C4a = (a4*delta*1000000000000)/(frequency_centre)

            L5a = (a5*delta*1000000000)/(frequency_centre)
            C5a = (1000000000000)/(delta*frequency_centre*a5)
            G6a = a6*output_resistance
```

```
            L1b = (1000000000)/(delta*frequency_centre*a1) #substitute C1 in low pass filter(Type
II)
            C1b = (a1*delta*1000000000000)/(frequency_centre) #substitute C1 in low pass
filter(Type II)
            L2b = (a2*delta*1000000000)/(frequency_centre) #substitute L2 in low pass filter(Type
II)
            C2b = (1000000000000)/(delta*frequency_centre*a2) #substitute L2 in low pass
filter(Type II)

            L3b = (1000000000)/(delta*frequency_centre*a3)
            C3b = (a3*delta*1000000000000)/(frequency_centre)
            L4b = (a4*delta*1000000000)/(frequency_centre)
            C4b = (1000000000000)/(delta*frequency_centre*a4)

            L5b = (1000000000)/(delta*frequency_centre*a5)
            C5b = (a5*delta*1000000000000)/(frequency_centre)
            G6b = a6*output_resistance

            print "\nType I Filter\n"
            print "L1 = " ,L1a, " (nH) "
            print "C1 = " ,C1a, " (pF) "
            print "L2 = " ,L2a, " (nH) "
            print "C2 = " ,C2a, " (pF) "

            print "L3 = " ,L3a, " (nH) "
            print "C3 = " ,C3a, " (pF) "
            print "L4 = " ,L4a, " (nH) "
            print "C4 = " ,C4a, " (pF) "

            print "L5 = " ,L5a, " (nH) "
            print "C5 = " ,C5a, " (pF) "
            print "G6 = " ,G6a, " (Ohm) "

            print "\nType II Filter\n"
            print "L1 = " ,L1b, " (nH) "
            print "C1 = " ,C1b, " (pF) "
            print "L2 = " ,L2b, " (nH) "
            print "C2 = " ,C2b, " (pF) "

            print "L3 = " ,L3b, " (nH) "
            print "C3 = " ,C3b, " (pF) "
            print "L4 = " ,L4b, " (nH) "
            print "C4 = " ,C4b, " (pF) "

            print "L5 = " ,L5b, " (nH) "
            print "C5 = " ,C5b, " (pF) "
            print "G6 = " ,G6b, " (Ohm) "

        elif filter_order==6:

            L1a = (a1*delta*1000000000)/(frequency_centre) #substitute L1 in low pass filter(Type I)
            C1a = (1000000000000)/(delta*frequency_centre*a1) #substitute L1 in low pass
filter(Type I)
            L2a = (1000000000)/(delta*frequency_centre*a2) #substitute C2 in low pass filter(Type I)
```

```
C2a = (a2*delta*1000000000000)/(frequency_centre) #substitute C2 in low pass filter(Type I)

L3a = (a3*delta*1000000000)/(frequency_centre)
C3a = (1000000000000)/(delta*frequency_centre*a3)
L4a = (1000000000)/(delta*frequency_centre*a4)
C4a = (a4*delta*1000000000000)/(frequency_centre)

L5a = (a5*delta*1000000000)/(frequency_centre)
C5a = (1000000000000)/(delta*frequency_centre*a5)
L6a = (1000000000)/(delta*frequency_centre*a6)
C6a = (a6*delta*1000000000000)/(frequency_centre)
G7a = a7*output_resistance

L1b = (1000000000)/(delta*frequency_centre*a1) #substitute C1 in low pass filter(Type II)
C1b = (a1*delta*1000000000000)/(frequency_centre) #substitute C1 in low pass filter(Type II)
L2b = (a2*delta*1000000000)/(frequency_centre) #substitute L2 in low pass filter(Type II)
C2b = (1000000000000)/(delta*frequency_centre*a2) #substitute L2 in low pass filter(Type II)

L3b = (1000000000)/(delta*frequency_centre*a3)
C3b = (a3*delta*1000000000000)/(frequency_centre)
L4b = (a4*delta*1000000000)/(frequency_centre)
C4b = (1000000000000)/(delta*frequency_centre*a4)

L5b = (1000000000)/(delta*frequency_centre*a5)
C5b = (a5*delta*1000000000000)/(frequency_centre)
L6b = (a6*delta*1000000000)/(frequency_centre)
C6b = (1000000000000)/(delta*frequency_centre*a6)
G7b = a7*output_resistance

print "\nType I Filter\n"
print "L1 = " ,L1a, " (nH) "
print "C1 = " ,C1a, " (pF) "
print "L2 = " ,L2a, " (nH) "
print "C2 = " ,C2a, " (pF) "

print "L3 = " ,L3a, " (nH) "
print "C3 = " ,C3a, " (pF) "
print "L4 = " ,L4a, " (nH) "
print "C4 = " ,C4a, " (pF) "

print "L5 = " ,L5a, " (nH) "
print "C5 = " ,C5a, " (pF) "
print "L6 = " ,L6a, " (nH) "
print "C6 = " ,C6a, " (pF) "
print "G7 = " ,G7a, " (Ohm) "

print "\nType II Filter\n"
print "L1 = " ,L1b, " (nH) "
print "C1 = " ,C1b, " (pF) "
```

```
            print "L2 = " ,L2b, " (nH) "
            print "C2 = " ,C2b, " (pF) "

            print "L3 = " ,L3b, " (nH) "
            print "C3 = " ,C3b, " (pF) "
            print "L4 = " ,L4b, " (nH) "
            print "C4 = " ,C4b, " (pF) "

            print "L5 = " ,L5b, " (nH) "
            print "C5 = " ,C5b, " (pF) "
            print "L6 = " ,L6b, " (nH) "
            print "C6 = " ,C6b, " (pF) "
            print "G7 = " ,G7b, " (Ohm) "

        elif filter_order==7:

            L1a = (a1*delta*1000000000)/(frequency_centre) #substitute L1 in low pass filter(Type I)
            C1a = (1000000000000)/(delta*frequency_centre*a1) #substitute L1 in low pass
filter(Type I)
            L2a = (1000000000)/(delta*frequency_centre*a2) #substitute C2 in low pass filter(Type I)
            C2a = (a2*delta*1000000000000)/(frequency_centre) #substitute C2 in low pass
filter(Type I)

            L3a = (a3*delta*1000000000)/(frequency_centre)
            C3a = (1000000000000)/(delta*frequency_centre*a3)
            L4a = (1000000000)/(delta*frequency_centre*a4)
            C4a = (a4*delta*1000000000000)/(frequency_centre)

            L5a = (a5*delta*1000000000)/(frequency_centre)
            C5a = (1000000000000)/(delta*frequency_centre*a5)
            L6a = (1000000000)/(delta*frequency_centre*a6)
            C6a = (a6*delta*1000000000000)/(frequency_centre)

            L7a = (a7*delta*1000000000)/(frequency_centre)
            C7a = (1000000000000)/(delta*frequency_centre*a7)
            G8a = a8*output_resistance

            L1b = (1000000000)/(delta*frequency_centre*a1) #substitute C1 in low pass filter(Type
II)
            C1b = (a1*delta*1000000000000)/(frequency_centre) #substitute C1 in low pass
filter(Type II)
            L2b = (a2*delta*1000000000)/(frequency_centre) #substitute L2 in low pass filter(Type
II)
            C2b = (1000000000000)/(delta*frequency_centre*a2) #substitute L2 in low pass
filter(Type II)

            L3b = (1000000000)/(delta*frequency_centre*a3)
            C3b = (a3*delta*1000000000000)/(frequency_centre)
            L4b = (a4*delta*1000000000)/(frequency_centre)
            C4b = (1000000000000)/(delta*frequency_centre*a4)

            L5b = (1000000000)/(delta*frequency_centre*a5)
            C5b = (a5*delta*1000000000000)/(frequency_centre)
            L6b = (a6*delta*1000000000)/(frequency_centre)
```

```
        C6b = (1000000000000)/(delta*frequency_centre*a6)

        L7b = (1000000000)/(delta*frequency_centre*a7)
        C7b = (a7*delta*1000000000000)/(frequency_centre)
        G8b = a8*output_resistance

        print "\nType I Filter\n"
        print "L1 = " ,L1a, " (nH) "
        print "C1 = " ,C1a, " (pF) "
        print "L2 = " ,L2a, " (nH) "
        print "C2 = " ,C2a, " (pF) "

        print "L3 = " ,L3a, " (nH) "
        print "C3 = " ,C3a, " (pF) "
        print "L4 = " ,L4a, " (nH) "
        print "C4 = " ,C4a, " (pF) "

        print "L5 = " ,L5a, " (nH) "
        print "C5 = " ,C5a, " (pF) "
        print "L6 = " ,L6a, " (nH) "
        print "C6 = " ,C6a, " (pF) "

        print "L7 = " ,L7a, " (nH) "
        print "C7 = " ,C7a, " (pF) "
        print "G8 = " ,G8a, " (Ohm) "

        print "\nType II Filter\n"
        print "L1 = " ,L1b, " (nH) "
        print "C1 = " ,C1b, " (pF) "
        print "L2 = " ,L2b, " (nH) "
        print "C2 = " ,C2b, " (pF) "

        print "L3 = " ,L3b, " (nH) "
        print "C3 = " ,C3b, " (pF) "
        print "L4 = " ,L4b, " (nH) "
        print "C4 = " ,C4b, " (pF) "

        print "L5 = " ,L5b, " (nH) "
        print "C5 = " ,C5b, " (pF) "
        print "L6 = " ,L6b, " (nH) "
        print "C6 = " ,C6b, " (pF) "

        print "L7 = " ,L7b, " (nH) "
        print "C7 = " ,C7b, " (pF) "
        print "G8 = " ,G8b, " (Ohm) "

    elif filter_order==8:

        L1a = (a1*delta*1000000000)/(frequency_centre) #substitute L1 in low pass filter(Type I)
        C1a = (1000000000000)/(delta*frequency_centre*a1) #substitute L1 in low pass
filter(Type I)
        L2a = (1000000000)/(delta*frequency_centre*a2) #substitute C2 in low pass filter(Type I)
        C2a = (a2*delta*1000000000000)/(frequency_centre) #substitute C2 in low pass
filter(Type I)
```

```
L3a = (a3*delta*1000000000)/(frequency_centre)
C3a = (1000000000000)/(delta*frequency_centre*a3)
L4a = (1000000000)/(delta*frequency_centre*a4)
C4a = (a4*delta*1000000000000)/(frequency_centre)

L5a = (a5*delta*1000000000)/(frequency_centre)
C5a = (1000000000000)/(delta*frequency_centre*a5)
L6a = (1000000000)/(delta*frequency_centre*a6)
C6a = (a6*delta*1000000000000)/(frequency_centre)

L7a = (a7*delta*1000000000)/(frequency_centre)
C7a = (1000000000000)/(delta*frequency_centre*a7)
L8a = (1000000000)/(delta*frequency_centre*a8)
C8a = (a8*delta*1000000000000)/(frequency_centre)
G9a = a9*output_resistance

L1b = (1000000000)/(delta*frequency_centre*a1) #substitute C1 in low pass filter(Type
II)
C1b = (a1*delta*1000000000000)/(frequency_centre) #substitute C1 in low pass
filter(Type II)
L2b = (a2*delta*1000000000)/(frequency_centre) #substitute L2 in low pass filter(Type
II)
C2b = (1000000000000)/(delta*frequency_centre*a2) #substitute L2 in low pass
filter(Type II)

L3b = (1000000000)/(delta*frequency_centre*a3)
C3b = (a3*delta*1000000000000)/(frequency_centre)
L4b = (a4*delta*1000000000)/(frequency_centre)
C4b = (1000000000000)/(delta*frequency_centre*a4)

L5b = (1000000000)/(delta*frequency_centre*a5)
C5b = (a5*delta*1000000000000)/(frequency_centre)
L6b = (a6*delta*1000000000)/(frequency_centre)
C6b = (1000000000000)/(delta*frequency_centre*a6)

L7b = (1000000000)/(delta*frequency_centre*a7)
C7b = (a7*delta*1000000000000)/(frequency_centre)
L8b = (a8*delta*1000000000)/(frequency_centre)
C8b = (1000000000000)/(delta*frequency_centre*a8)
G9b = a9*output_resistance

print "\nType I Filter\n"
print "L1 = " ,L1a, " (nH) "
print "C1 = " ,C1a, " (pF) "
print "L2 = " ,L2a, " (nH) "
print "C2 = " ,C2a, " (pF) "

print "L3 = " ,L3a, " (nH) "
print "C3 = " ,C3a, " (pF) "
print "L4 = " ,L4a, " (nH) "
print "C4 = " ,C4a, " (pF) "

print "L5 = " ,L5a, " (nH) "
```

```
            print "C5 = " ,C5a, " (pF) "
            print "L6 = " ,L6a, " (nH) "
            print "C6 = " ,C6a, " (pF) "

            print "L7 = " ,L7a, " (nH) "
            print "C7 = " ,C7a, " (pF) "
            print "L8 = " ,L8a, " (nH) "
            print "C8 = " ,C8a, " (pF) "
            print "G9 = " ,G9a, " (Ohm) "

            print "\nType II Filter\n"
            print "L1 = " ,L1b, " (nH) "
            print "C1 = " ,C1b, " (pF) "
            print "L2 = " ,L2b, " (nH) "
            print "C2 = " ,C2b, " (pF) "

            print "L3 = " ,L3b, " (nH) "
            print "C3 = " ,C3b, " (pF) "
            print "L4 = " ,L4b, " (nH) "
            print "C4 = " ,C4b, " (pF) "

            print "L5 = " ,L5b, " (nH) "
            print "C5 = " ,C5b, " (pF) "
            print "L6 = " ,L6b, " (nH) "
            print "C6 = " ,C6b, " (pF) "

            print "L7 = " ,L7b, " (nH) "
            print "C7 = " ,C7b, " (pF) "
            print "L8 = " ,L8b, " (nH) "
            print "C8 = " ,C8b, " (pF) "
            print "G9 = " ,G9b, " (Ohm) "

        elif filter_order==9:

            L1a = (a1*delta*1000000000)/(frequency_centre) #substitute L1 in low pass filter(Type I)
            C1a = (1000000000000)/(delta*frequency_centre*a1) #substitute L1 in low pass
filter(Type I)
            L2a = (1000000000)/(delta*frequency_centre*a2) #substitute C2 in low pass filter(Type I)
            C2a = (a2*delta*1000000000000)/(frequency_centre) #substitute C2 in low pass
filter(Type I)

            L3a = (a3*delta*1000000000)/(frequency_centre)
            C3a = (1000000000000)/(delta*frequency_centre*a3)
            L4a = (1000000000)/(delta*frequency_centre*a4)
            C4a = (a4*delta*1000000000000)/(frequency_centre)

            L5a = (a5*delta*1000000000)/(frequency_centre)
            C5a = (1000000000000)/(delta*frequency_centre*a5)
            L6a = (1000000000)/(delta*frequency_centre*a6)
            C6a = (a6*delta*1000000000000)/(frequency_centre)

            L7a = (a7*delta*1000000000)/(frequency_centre)
            C7a = (1000000000000)/(delta*frequency_centre*a7)
            L8a = (1000000000)/(delta*frequency_centre*a8)
```

```
C8a = (a8*delta*1000000000000)/(frequency_centre)

L9a = (a9*delta*1000000000)/(frequency_centre)
C9a = (1000000000000)/(delta*frequency_centre*a9)
G10a = a10*output_resistance

L1b = (1000000000)/(delta*frequency_centre*a1) #substitute C1 in low pass filter(Type
II)
C1b = (a1*delta*1000000000000)/(frequency_centre) #substitute C1 in low pass
filter(Type II)
L2b = (a2*delta*1000000000)/(frequency_centre) #substitute L2 in low pass filter(Type
II)
C2b = (1000000000000)/(delta*frequency_centre*a2) #substitute L2 in low pass
filter(Type II)

L3b = (1000000000)/(delta*frequency_centre*a3)
C3b = (a3*delta*1000000000000)/(frequency_centre)
L4b = (a4*delta*1000000000)/(frequency_centre)
C4b = (1000000000000)/(delta*frequency_centre*a4)

L5b = (1000000000)/(delta*frequency_centre*a5)
C5b = (a5*delta*1000000000000)/(frequency_centre)
L6b = (a6*delta*1000000000)/(frequency_centre)
C6b = (1000000000000)/(delta*frequency_centre*a6)

L7b = (1000000000)/(delta*frequency_centre*a7)
C7b = (a7*delta*1000000000000)/(frequency_centre)
L8b = (a8*delta*1000000000)/(frequency_centre)
C8b = (1000000000000)/(delta*frequency_centre*a8)

L9b = (1000000000)/(delta*frequency_centre*a9)
C9b = (a9*delta*1000000000000)/(frequency_centre)
G10b = a10*output_resistance

print "\nType I Filter\n"
print "L1 = " ,L1a, " (nH) "
print "C1 = " ,C1a, " (pF) "
print "L2 = " ,L2a, " (nH) "
print "C2 = " ,C2a, " (pF) "

print "L3 = " ,L3a, " (nH) "
print "C3 = " ,C3a, " (pF) "
print "L4 = " ,L4a, " (nH) "
print "C4 = " ,C4a, " (pF) "

print "L5 = " ,L5a, " (nH) "
print "C5 = " ,C5a, " (pF) "
print "L6 = " ,L6a, " (nH) "
print "C6 = " ,C6a, " (pF) "

print "L7 = " ,L7a, " (nH) "
print "C7 = " ,C7a, " (pF) "
print "L8 = " ,L8a, " (nH) "
print "C8 = " ,C8a, " (pF) "
```

```
L9a = (a9*delta*1000000000)/(frequency_centre)
C9a = (1000000000000)/(delta*frequency_centre*a9)
L10a = (1000000000)/(delta*frequency_centre*a10)
C10a = (a10*delta*1000000000000)/(frequency_centre)
G11a = a11*output_resistance

L1b = (1000000000)/(delta*frequency_centre*a1) #substitute C1 in low pass filter(Type
II)
C1b = (a1*delta*1000000000000)/(frequency_centre) #substitute C1 in low pass
filter(Type II)
L2b = (a2*delta*1000000000)/(frequency_centre) #substitute L2 in low pass filter(Type
II)
C2b = (1000000000000)/(delta*frequency_centre*a2) #substitute L2 in low pass
filter(Type II)

L3b = (1000000000)/(delta*frequency_centre*a3)
C3b = (a3*delta*1000000000000)/(frequency_centre)
L4b = (a4*delta*1000000000)/(frequency_centre)
C4b = (1000000000000)/(delta*frequency_centre*a4)

L5b = (1000000000)/(delta*frequency_centre*a5)
C5b = (a5*delta*1000000000000)/(frequency_centre)
L6b = (a6*delta*1000000000)/(frequency_centre)
C6b = (1000000000000)/(delta*frequency_centre*a6)

L7b = (1000000000)/(delta*frequency_centre*a7)
C7b = (a7*delta*1000000000000)/(frequency_centre)
L8b = (a8*delta*1000000000)/(frequency_centre)
C8b = (1000000000000)/(delta*frequency_centre*a8)

L9b = (1000000000)/(delta*frequency_centre*a9)
C9b = (a9*delta*1000000000000)/(frequency_centre)
L10b = (a10*delta*1000000000)/(frequency_centre)
C10b = (1000000000000)/(delta*frequency_centre*a10)
G11b = a11*output_resistance

print "\nType I Filter\n"
print "L1 = " ,L1a, " (nH) "
print "C1 = " ,C1a, " (pF) "
print "L2 = " ,L2a, " (nH) "
print "C2 = " ,C2a, " (pF) "

print "L3 = " ,L3a, " (nH) "
print "C3 = " ,C3a, " (pF) "
print "L4 = " ,L4a, " (nH) "
print "C4 = " ,C4a, " (pF) "

print "L5 = " ,L5a, " (nH) "
print "C5 = " ,C5a, " (pF) "
print "L6 = " ,L6a, " (nH) "
print "C6 = " ,C6a, " (pF) "

print "L7 = " ,L7a, " (nH) "
print "C7 = " ,C7a, " (pF) "
```

```
            print "L8 = " ,L8a, " (nH) "
            print "C8 = " ,C8a, " (pF) "

            print "L9 = " ,L9a, " (nH) "
            print "C9 = " ,C9a, " (pF) "
            print "L10 = " ,L10a, " (nH) "
            print "C10 = " ,C10a, " (pF) "
            print "G11 = " ,G11a, " (Ohm) "

            print "\nType II Filter\n"
            print "L1 = " ,L1b, " (nH) "
            print "C1 = " ,C1b, " (pF) "
            print "L2 = " ,L2b, " (nH) "
            print "C2 = " ,C2b, " (pF) "

            print "L3 = " ,L3b, " (nH) "
            print "C3 = " ,C3b, " (pF) "
            print "L4 = " ,L4b, " (nH) "
            print "C4 = " ,C4b, " (pF) "

            print "L5 = " ,L5b, " (nH) "
            print "C5 = " ,C5b, " (pF) "
            print "L6 = " ,L6b, " (nH) "
            print "C6 = " ,C6b, " (pF) "

            print "L7 = " ,L7b, " (nH) "
            print "C7 = " ,C7b, " (pF) "
            print "L8 = " ,L8b, " (nH) "
            print "C8 = " ,C8b, " (pF) "

            print "L9 = " ,L9b, " (nH) "
            print "C9 = " ,C9b, " (pF) "
            print "L10 = " ,L10b, " (nH) "
            print "C10 = " ,C10b, " (pF) "
            print "G11 = " ,G11b, " (Ohm) "

        else:
            print "Please enter filter order between 1 to 10"
    else:
        print "Please enter filter type between 1 to 4"

else:
   print "Wrong pie value"

#########################################################################
#
#
#          End of Program
#
#
#
#########################################################################
```

APPENDIX B

PLOTTING CODE FOR 1ST ORDER FILTER

```
from numpy import *
from matplotlib.pyplot import *
from math import sqrt


######### Taking input from user

pi=3.141592654
L1a=input("Enter the Inductor, L1 value: ")
output_resistance=input("Enter the Output Resistance(ohm) desired: ")


######## 0.1GHZ freq

######## ABCD matrix calculation

A1_0_1 = 1
B1_0_1 = (1j*2*pi*0.1e12*L1a)
C1_0_1 = 0
D1_0_1 = 1


W_0_1=A1_0_1
X_0_1=B1_0_1
Y_0_1=C1_0_1
Z_0_1=D1_0_1

######## ABCD parameter is then converted to S parameter

S11_0_1 = (W_0_1 + (X_0_1/output_resistance) - (Y_0_1*output_resistance) - Z_0_1)/(W_0_1 +
(X_0_1/output_resistance) + (Y_0_1*output_resistance) + Z_0_1)
S12_0_1 = (2*((W_0_1*Z_0_1)-(X_0_1*Y_0_1)))/(W_0_1 + (X_0_ 1/output_resistance) +
(Y_0_1*output_resistance) + Z_0_1)
S21_0_1 = 2/(W_0_1 + (X_0_1/output_resistance) + (Y_0_1*output_resistance) + Z_0_1)
S22_0_1 = (- W_0_1 + (X_0_1/output_resistance) - (Y_0_1*output_resistance) + Z_0_1)/(W_0_1 +
(X_0_1/output_resistance) + (Y_0_1*output_resistance) + Z_0_1)


S12amp_0_1=sqrt((S12_0_1.real*S12_0_1.real)+(S12_0_1.imag*S12_0_1.imag))



######## 1GHZ freq

A1_1 = 1
B1_1 = (1j*2*pi*1e12*L1a)
C1_1 = 0
D1_1 = 1
```

```
W_1=A1_1
X_1=B1_1
Y_1=C1_1
Z_1=D1_1


S11_1 = (W_1 + (X_1/output_resistance) - (Y_1*output_resistance) - Z_1)/(W_1 +
(X_1/output_resistance) + (Y_1*output_resistance) + Z_1)
S12_1 = (2*((W_1*Z_1)-(X_1*Y_1)))/(W_1 + (X_1/output_resistance) + (Y_1*output_resistance) +
Z_1)
S21_1 = 2/(W_1 + (X_1/output_resistance) + (Y_1*output_resistance) + Z_1)
S22_1 = (- W_1 + (X_1/output_resistance) - (Y_1*output_resistance) + Z_1)/(W_1 +
(X_1/output_resistance) + (Y_1*output_resistance) + Z_1)


S12amp_1=sqrt((S12_1.real*S12_1.real)+(S12_1.imag*S12_1.imag))


####### 2GHZ freq

A1_2 = 1
B1_2 = (1j*2*pi*2e12*L1a)
C1_2 = 0
D1_2 = 1

W_2=A1_2
X_2=B1_2
Y_2=C1_2
Z_2=D1_2


S11_2 = (W_2 + (X_2/output_resistance) - (Y_2*output_resistance) - Z_2)/(W_2 +
(X_2/output_resistance) + (Y_2*output_resistance) + Z_2)
S12_2 = (2*((W_2*Z_2)-(X_2*Y_2)))/(W_2 + (X_2/output_resistance) + (Y_2*output_resistance) +
Z_2)
S21_2 = 2/(W_2 + (X_2/output_resistance) + (Y_2*output_resistance) + Z_2)
S22_2 = (- W_2 + (X_2/output_resistance) - (Y_2*output_resistance) + Z_2)/(W_2 +
(X_2/output_resistance) + (Y_2*output_resistance) + Z_2)


S12amp_2=sqrt((S12_2.real*S12_2.real)+(S12_2.imag*S12_2.imag))


####### 3GHZ freq

A1_3 = 1
B1_3 = (1j*2*pi*3e12*L1a)
C1_3 = 0
D1_3 = 1

W_3=A1_3
X_3=B1_3
Y_3=C1_3
```

```
Z_3=D1_3


S11_3 = (W_3 + (X_3/output_resistance) - (Y_3*output_resistance) - Z_3)/(W_3 +
(X_3/output_resistance) + (Y_3*output_resistance) + Z_3)
S12_3 = (2*((W_3*Z_3)-(X_3*Y_3)))/(W_3 + (X_3/output_resistance) + (Y_3*output_resistance) +
Z_3)
S21_3 = 2/(W_3 + (X_3/output_resistance) + (Y_3*output_resistance) + Z_3)
S22_3 = (- W_3 + (X_3/output_resistance) - (Y_3*output_resistance) + Z_3)/(W_3 +
(X_3/output_resistance) + (Y_3*output_resistance) + Z_3)


S12amp_3=sqrt((S12_3.real*S12_3.real)+(S12_3.imag*S12_3.imag))


####### 4GHZ freq


A1_4 = 1
B1_4 = (1j*2*pi*4e12*L1a)
C1_4 = 0
D1_4 = 1

W_4=A1_4
X_4=B1_4
Y_4=C1_4
Z_4=D1_4


S11_4 = (W_4 + (X_4/output_resistance) - (Y_4*output_resistance) - Z_4)/(W_4 +
(X_4/output_resistance) + (Y_4*output_resistance) + Z_4)
S12_4 = (2*((W_4*Z_4)-(X_4*Y_4)))/(W_4 + (X_4/output_resistance) + (Y_4*output_resistance) +
Z_4)
S21_4 = 2/(W_4 + (X_4/output_resistance) + (Y_4*output_resistance) + Z_4)
S22_4 = (- W_4 + (X_4/output_resistance) - (Y_4*output_resistance) + Z_4)/(W_4 +
(X_4/output_resistance) + (Y_4*output_resistance) + Z_4)


S12amp_4=sqrt((S12_4.real*S12_4.real)+(S12_4.imag*S12_4.imag))


####### 5GHZ freq


A1_5 = 1
B1_5 = (1j*2*pi*5e12*L1a)
C1_5 = 0
D1_5 = 1

W_5=A1_5
X_5=B1_5
Y_5=C1_5
Z_5=D1_5
```

```
S11_5 = (W_5 + (X_5/output_resistance) - (Y_5*output_resistance) - Z_5)/(W_5 +
(X_5/output_resistance) + (Y_5*output_resistance) + Z_5)
S12_5 = (2*((W_5*Z_5)-(X_5*Y_5)))/(W_5 + (X_5/output_resistance) + (Y_5*output_resistance) +
Z_5)
S21_5 = 2/(W_5 + (X_5/output_resistance) + (Y_5*output_resistance) + Z_5)
S22_5 = (- W_5 + (X_5/output_resistance) - (Y_5*output_resistance) + Z_5)/(W_5 +
(X_5/output_resistance) + (Y_5*output_resistance) + Z_5)


S12amp_5=sqrt((S12_5.real*S12_5.real)+(S12_5.imag*S12_5.imag))


####### 6GHZ freq

A1_6 = 1
B1_6 = (1j*2*pi*6e12*L1a)
C1_6 = 0
D1_6 = 1

W_6=A1_6
X_6=B1_6
Y_6=C1_6
Z_6=D1_6


S11_6 = (W_6 + (X_6/output_resistance) - (Y_6*output_resistance) - Z_6)/(W_6 +
(X_6/output_resistance) + (Y_6*output_resistance) + Z_6)
S12_6 = (2*((W_6*Z_6)-(X_6*Y_6)))/(W_6 + (X_6/output_resistance) + (Y_6*output_resistance) +
Z_6)
S21_6 = 2/(W_6 + (X_6/output_resistance) + (Y_6*output_resistance) + Z_6)
S22_6 = (- W_6 + (X_6/output_resistance) - (Y_6*output_resistance) + Z_6)/(W_6 +
(X_6/output_resistance) + (Y_6*output_resistance) + Z_6)


S12amp_6=sqrt((S12_6.real*S12_6.real)+(S12_6.imag*S12_6.imag))


####### 7GHZ freq



A1_7 = 1
B1_7 = (1j*2*pi*7e12*L1a)
C1_7 = 0
D1_7 = 1

W_7=A1_7
X_7=B1_7
Y_7=C1_7
Z_7=D1_7
```

```
S11_7 = (W_7 + (X_7/output_resistance) - (Y_7*output_resistance) - Z_7)/(W_7 +
(X_7/output_resistance) + (Y_7*output_resistance) + Z_7)
S12_7 = (2*((W_7*Z_7)-(X_7*Y_7)))/(W_7 + (X_7/output_resistance) + (Y_7*output_resistance) +
Z_7)
S21_7 = 2/(W_7 + (X_7/output_resistance) + (Y_7*output_resistance) + Z_7)
S22_7 = (- W_7 + (X_7/output_resistance) - (Y_7*output_resistance) + Z_7)/(W_7 +
(X_7/output_resistance) + (Y_7*output_resistance) + Z_7)


S12amp_7=sqrt((S12_7.real*S12_7.real)+(S12_7.imag*S12_7.imag))


######## 8GHZ freq


A1_8 = 1
B1_8 = (1j*2*pi*8e12*L1a)
C1_8 = 0
D1_8 = 1

W_8=A1_8
X_8=B1_8
Y_8=C1_8
Z_8=D1_8


S11_8 = (W_8 + (X_8/output_resistance) - (Y_8*output_resistance) - Z_8)/(W_8 +
(X_8/output_resistance) + (Y_8*output_resistance) + Z_8)
S12_8 = (2*((W_8*Z_8)-(X_8*Y_8)))/(W_8 + (X_8/output_resistance) + (Y_8*output_resistance) +
Z_8)
S21_8 = 2/(W_8 + (X_8/output_resistance) + (Y_8*output_resistance) + Z_8)
S22_8 = (- W_8 + (X_8/output_resistance) - (Y_8*output_resistance) + Z_8)/(W_8 +
(X_8/output_resistance) + (Y_8*output_resistance) + Z_8)


S12amp_8=sqrt((S12_8.real*S12_8.real)+(S12_8.imag*S12_8.imag))


######## 9GHZ freq



A1_9 = 1
B1_9 = (1j*2*pi*9e12*L1a)
C1_9 = 0
D1_9 = 1

W_9=A1_9
X_9=B1_9
Y_9=C1_9
Z_9=D1_9
```

```
S11_9 = (W_9 + (X_9/output_resistance) - (Y_9*output_resistance) - Z_9)/(W_9 +
(X_9/output_resistance) + (Y_9*output_resistance) + Z_9)
S12_9 = (2*((W_9*Z_9)-(X_9*Y_9)))/(W_9 + (X_9/output_resistance) + (Y_9*output_resistance) +
Z_9)
S21_9 = 2/(W_9 + (X_9/output_resistance) + (Y_9*output_resistance) + Z_9)
S22_9 = (- W_9 + (X_9/output_resistance) - (Y_9*output_resistance) + Z_9)/(W_9 +
(X_9/output_resistance) + (Y_9*output_resistance) + Z_9)


S12amp_9=sqrt((S12_9.real*S12_9.real)+(S12_9.imag*S12_9.imag))


####### 10GHZ freq


A1_10 = 1
B1_10 = (1j*2*pi*10e12*L1a)
C1_10 = 0
D1_10 = 1

W_10=A1_10
X_10=B1_10
Y_10=C1_10
Z_10=D1_10


S11_10 = (W_10 + (X_10/output_resistance) - (Y_10*output_resistance) - Z_10)/(W_10 +
(X_10/output_resistance) + (Y_10*output_resistance) + Z_10)
S12_10 = (2*((W_10*Z_10)-(X_10*Y_10)))/(W_10 + (X_10/output_resistance) +
(Y_10*output_resistance) + Z_10)
S21_10 = 2/(W_10 + (X_10/output_resistance) + (Y_10*output_resistance) + Z_10)
S22_10 = (- W_10 + (X_10/output_resistance) - (Y_10*output_resistance) + Z_10)/(W_10 +
(X_10/output_resistance) + (Y_10*output_resistance) + Z_10)


S12amp_10=sqrt((S12_10.real*S12_10.real)+(S12_10.imag*S12_10.imag))


####### end

########## plotting the graph

figure()
y=array([S12amp_0_1, S12amp_1, S12amp_2, S12amp_3, S12amp_4, S12amp_5, S12amp_6,
S12amp_7, S12amp_8, S12amp_9, S12amp_10])
t=array([0.1,1,2,3,4,5,6,7,8,9,10])
plot(t,y, 'x:')
show()
```

APPENDIX C

PLOTTING CODE FOR 2ND ORDER FILTER

```
from numpy import *
from matplotlib.pyplot import *
from math import sqrt

######### Taking input from user

pi=3.141592654
L1a=input("Enter the Inductor, L1 value: ")
C2a=input("Enter the Capacitor, C2 value: ")
output_resistance=input("Enter the Output Resistance(ohm) desired: ")

####### 0.1GHZ freq

####### ABCD matrix calculation

A1_0_1 = 1
B1_0_1 = (1j*2*pi*0.1e12*L1a)
C1_0_1 = 0
D1_0_1 = 1

A2_0_1 = 1
B2_0_1 = 0
C2_0_1 = 1/(1j*2*pi*0.1e12*C2a)
D2_0_1 = 1

####### ABCD parameter for the system cascaded

W_0_1=(A1_0_1*A2_0_1)+(B1_0_1*C2_0_1)
X_0_1=(A1_0_1*B2_0_1)+(B1_0_1*D2_0_1)
Y_0_1=(C1_0_1*A2_0_1)+(D1_0_1*C2_0_1)
Z_0_1=(C1_0_1*B2_0_1)+(D1_0_1*D2_0_1)

####### ABCD parameter is then converted to S parameter

S11_0_1 = (W_0_1 + (X_0_1/output_resistance) - (Y_0_1*output_resistance) - Z_0_1)/(W_0_1 +
(X_0_1/output_resistance) + (Y_0_1*output_resistance) + Z_0_1)
S12_0_1 = (2*((W_0_1*Z_0_1)-(X_0_1*Y_0_1)))/(W_0_1 + (X_0_1/output_resistance) +
(Y_0_1*output_resistance) + Z_0_1)
S21_0_1 = 2/(W_0_1 + (X_0_1/output_resistance) + (Y_0_1*output_resistance) + Z_0_1)
S22_0_1 = (- W_0_1 + (X_0_1/output_resistance) - (Y_0_1*output_resistance) + Z_0_1)/(W_0_1 +
(X_0_1/output_resistance) + (Y_0_1*output_resistance) + Z_0_1)

S12amp_0_1=sqrt((S12_0_1.real*S12_0_1.real)+(S12_0_1.imag*S12_0_1.imag))
```

```
####### 1GHZ freq

A1_1 = 1
B1_1 = (1j*2*pi*1e12*L1a)
C1_1 = 0
D1_1 = 1

A2_1 = 1
B2_1 = 0
C2_1 = 1/(1j*2*pi*1e12*C2a)
D2_1 = 1


W_1=(A1_1*A2_1)+(B1_1*C2_1)
X_1=(A1_1*B2_1)+(B1_1*D2_1)
Y_1=(C1_1*A2_1)+(D1_1*C2_1)
Z_1=(C1_1*B2_1)+(D1_1*D2_1)


S11_1 = (W_1 + (X_1/output_resistance) - (Y_1*output_resistance) - Z_1)/(W_1 +
(X_1/output_resistance) + (Y_1*output_resistance) + Z_1)
S12_1 = (2*((W_1*Z_1)-(X_1*Y_1)))/(W_1 + (X_1/output_resistance) + (Y_1*output_resistance) +
Z_1)
S21_1 = 2/(W_1 + (X_1/output_resistance) + (Y_1*output_resistance) + Z_1)
S22_1 = (- W_1 + (X_1/output_resistance) - (Y_1*output_resistance) + Z_1)/(W_1 +
(X_1/output_resistance) + (Y_1*output_resistance) + Z_1)


S12amp_1=sqrt((S12_1.real*S12_1.real)+(S12_1.imag*S12_1.imag))


####### 2GHZ freq

A1_2 = 1
B1_2 = (1j*2*pi*2e12*L1a)
C1_2 = 0
D1_2 = 1

A2_2 = 1
B2_2 = 0
C2_2 = 1/(1j*2*pi*2e12*C2a)
D2_2 = 1


W_2=(A1_2*A2_2)+(B1_2*C2_2)
X_2=(A1_2*B2_2)+(B1_2*D2_2)
Y_2=(C1_2*A2_2)+(D1_2*C2_2)
Z_2=(C1_2*B2_2)+(D1_2*D2_2)


S11_2 = (W_2 + (X_2/output_resistance) - (Y_2*output_resistance) - Z_2)/(W_2 +
(X_2/output_resistance) + (Y_2*output_resistance) + Z_2)
S12_2 = (2*((W_2*Z_2)-(X_2*Y_2)))/(W_2 + (X_2/output_resistance) + (Y_2*output_resistance) +
Z_2)
```

```
S21_2 = 2/(W_2 + (X_2/output_resistance) + (Y_2*output_resistance) + Z_2)
S22_2 = (- W_2 + (X_2/output_resistance) - (Y_2*output_resistance) + Z_2)/(W_2 +
(X_2/output_resistance) + (Y_2*output_resistance) + Z_2)


S12amp_2=sqrt((S12_2.real*S12_2.real)+(S12_2.imag*S12_2.imag))


####### 3GHZ freq

A1_3 = 1
B1_3 = (1j*2*pi*3e12*L1a)
C1_3 = 0
D1_3 = 1

A2_3 = 1
B2_3 = 0
C2_3 = 1/(1j*2*pi*3e12*C2a)
D2_3 = 1


W_3=(A1_3*A2_3)+(B1_3*C2_3)
X_3=(A1_3*B2_3)+(B1_3*D2_3)
Y_3=(C1_3*A2_3)+(D1_3*C2_3)
Z_3=(C1_3*B2_3)+(D1_3*D2_3)


S11_3 = (W_3 + (X_3/output_resistance) - (Y_3*output_resistance) - Z_3)/(W_3 +
(X_3/output_resistance) + (Y_3*output_resistance) + Z_3)
S12_3 = (2*((W_3*Z_3)-(X_3*Y_3)))/(W_3 + (X_3/output_resistance) + (Y_3*output_resistance) +
Z_3)
S21_3 = 2/(W_3 + (X_3/output_resistance) + (Y_3*output_resistanc :) + Z_3)
S22_3 = (- W_3 + (X_3/output_resistance) - (Y_3*output_resistance) + Z_3)/(W_3 +
(X_3/output_resistance) + (Y_3*output_resistance) + Z_3)


S12amp_3=sqrt((S12_3.real*S12_3.real)+(S12_3.imag*S12_3.imag))


####### 4GHZ freq


A1_4 = 1
B1_4 = (1j*2*pi*4e12*L1a)
C1_4 = 0
D1_4 = 1

A2_4 = 1
B2_4 = 0
C2_4 = 1/(1j*2*pi*4e12*C2a)
D2_4 = 1


W_4=(A1_4*A2_4)+(B1_4*C2_4)
```

```
X_4=(A1_4*B2_4)+(B1_4*D2_4)
Y_4=(C1_4*A2_4)+(D1_4*C2_4)
Z_4=(C1_4*B2_4)+(D1_4*D2_4)


S11_4 = (W_4 + (X_4/output_resistance) - (Y_4*output_resistance) - Z_4)/(W_4 +
(X_4/output_resistance) + (Y_4*output_resistance) + Z_4)
S12_4 = (2*((W_4*Z_4)-(X_4*Y_4)))/(W_4 + (X_4/output_resistance) + (Y_4*output_resistance) +
Z_4)
S21_4 = 2/(W_4 + (X_4/output_resistance) + (Y_4*output_resistance) + Z_4)
S22_4 = (- W_4 + (X_4/output_resistance) - (Y_4*output_resistance) + Z_4)/(W_4 +
(X_4/output_resistance) + (Y_4*output_resistance) + Z_4)


S12amp_4=sqrt((S12_4.real*S12_4.real)+(S12_4.imag*S12_4.imag))


####### 5GHZ freq


A1_5 = 1
B1_5 = (1j*2*pi*5e12*L1a)
C1_5 = 0
D1_5 = 1

A2_5 = 1
B2_5 = 0
C2_5 = 1/(1j*2*pi*5e12*C2a)
D2_5 = 1


W_5=(A1_5*A2_5)+(B1_5*C2_5)
X_5=(A1_5*B2_5)+(B1_5*D2_5)
Y_5=(C1_5*A2_5)+(D1_5*C2_5)
Z_5=(C1_5*B2_5)+(D1_5*D2_5)


S11_5 = (W_5 + (X_5/output_resistance) - (Y_5*output_resistance) - Z_5)/(W_5 +
(X_5/output_resistance) + (Y_5*output_resistance) + Z_5)
S12_5 = (2*((W_5*Z_5)-(X_5*Y_5)))/(W_5 + (X_5/output_resistance) + (Y_5*output_resistance) +
Z_5)
S21_5 = 2/(W_5 + (X_5/output_resistance) + (Y_5*output_resistance) + Z_5)
S22_5 = (- W_5 + (X_5/output_resistance) - (Y_5*output_resistance) + Z_5)/(W_5 +
(X_5/output_resistance) + (Y_5*output_resistance) + Z_5)


S12amp_5=sqrt((S12_5.real*S12_5.real)+(S12_5.imag*S12_5.imag))


####### 6GHZ freq

A1_6 = 1
B1_6 = (1j*2*pi*6e12*L1a)
C1_6 = 0
```

```
D1_6 = 1

A2_6 = 1
B2_6 = 0
C2_6 = 1/(1j*2*pi*6e12*C2a)
D2_6 = 1


W_6=(A1_6*A2_6)+(B1_6*C2_6)
X_6=(A1_6*B2_6)+(B1_6*D2_6)
Y_6=(C1_6*A2_6)+(D1_6*C2_6)
Z_6=(C1_6*B2_6)+(D1_6*D2_6)


S11_6 = (W_6 + (X_6/output_resistance) - (Y_6*output_resistance) - Z_6)/(W_6 +
(X_6/output_resistance) + (Y_6*output_resistance) + Z_6)
S12_6 = (2*((W_6*Z_6)-(X_6*Y_6)))/(W_6 + (X_6/output_resistance) + (Y_6*output_resistance) +
Z_6)
S21_6 = 2/(W_6 + (X_6/output_resistance) + (Y_6*output_resistance) + Z_6)
S22_6 = (- W_6 + (X_6/output_resistance) - (Y_6*output_resistance) + Z_6)/(W_6 +
(X_6/output_resistance) + (Y_6*output_resistance) + Z_6)


S12amp_6=sqrt((S12_6.real*S12_6.real)+(S12_6.imag*S12_6.imag))


####### 7GHZ freq



A1_7 = 1
B1_7 = (1j*2*pi*7e12*L1a)
C1_7 = 0
D1_7 = 1

A2_7 = 1
B2_7 = 0
C2_7 = 1/(1j*2*pi*7e12*C2a)
D2_7 = 1


W_7=(A1_7*A2_7)+(B1_7*C2_7)
X_7=(A1_7*B2_7)+(B1_7*D2_7)
Y_7=(C1_7*A2_7)+(D1_7*C2_7)
Z_7=(C1_7*B2_7)+(D1_7*D2_7)


S11_7 = (W_7 + (X_7/output_resistance) - (Y_7*output_resistance) - Z_7)/(W_7 +
(X_7/output_resistance) + (Y_7*output_resistance) + Z_7)
S12_7 = (2*((W_7*Z_7)-(X_7*Y_7)))/(W_7 + (X_7/output_resistance) + (Y_7*output_resistance) +
Z_7)
S21_7 = 2/(W_7 + (X_7/output_resistance) + (Y_7*output_resistance) + Z_7)
S22_7 = (- W_7 + (X_7/output_resistance) - (Y_7*output_resistance) + Z_7)/(W_7 +
(X_7/output_resistance) + (Y_7*output_resistance) + Z_7)
```

```
S12amp_7=sqrt((S12_7.real*S12_7.real)+(S12_7.imag*S12_7.imag))


######## 8GHZ freq


A1_8 = 1
B1_8 = (1j*2*pi*8e12*L1a)
C1_8 = 0
D1_8 = 1

A2_8 = 1
B2_8 = 0
C2_8 = 1/(1j*2*pi*8e12*C2a)
D2_8 = 1


W_8=(A1_8*A2_8)+(B1_8*C2_8)
X_8=(A1_8*B2_8)+(B1_8*D2_8)
Y_8=(C1_8*A2_8)+(D1_8*C2_8)
Z_8=(C1_8*B2_8)+(D1_8*D2_8)


S11_8 = (W_8 + (X_8/output_resistance) - (Y_8*output_resistance) - Z_8)/(W_8 +
(X_8/output_resistance) + (Y_8*output_resistance) + Z_8)
S12_8 = (2*((W_8*Z_8)-(X_8*Y_8)))/(W_8 + (X_8/output_resistance) + (Y_8*output_resistance) +
Z_8)
S21_8 = 2/(W_8 + (X_8/output_resistance) + (Y_8*output_resistance) + Z_8)
S22_8 = (- W_8 + (X_8/output_resistance) - (Y_8*output_resistance) + Z_8)/(W_8 +
(X_8/output_resistance) + (Y_8*output_resistance) + Z_8)


S12amp_8=sqrt((S12_8.real*S12_8.real)+(S12_8.imag*S12_8.imag))


######## 9GHZ freq



A1_9 = 1
B1_9 = (1j*2*pi*9e12*L1a)
C1_9 = 0
D1_9 = 1

A2_9 = 1
B2_9 = 0
C2_9 = 1/(1j*2*pi*9e12*C2a)
D2_9 = 1


W_9=(A1_9*A2_9)+(B1_9*C2_9)
X_9=(A1_9*B2_9)+(B1_9*D2_9)
```

```
Y_9=(C1_9*A2_9)+(D1_9*C2_9)
Z_9=(C1_9*B2_9)+(D1_9*D2_9)


S11_9 = (W_9 + (X_9/output_resistance) - (Y_9*output_resistance) - Z_9)/(W_9 +
(X_9/output_resistance) + (Y_9*output_resistance) + Z_9)
S12_9 = (2*((W_9*Z_9)-(X_9*Y_9)))/(W_9 + (X_9/output_resistance) + (Y_9*output_resistance) +
Z_9)
S21_9 = 2/(W_9 + (X_9/output_resistance) + (Y_9*output_resistance) + Z_9)
S22_9 = (- W_9 + (X_9/output_resistance) - (Y_9*output_resistance) + Z_9)/(W_9 +
(X_9/output_resistance) + (Y_9*output_resistance) + Z_9)


S12amp_9=sqrt((S12_9.real*S12_9.real)+(S12_9.imag*S12_9.imag))


####### 10GHZ freq


A1_10 = 1
B1_10 = (1j*2*pi*10e12*L1a)
C1_10 = 0
D1_10 = 1

A2_10 = 1
B2_10 = 0
C2_10 = 1/(1j*2*pi*10e12*C2a)
D2_10 = 1


W_10=(A1_10*A2_10)+(B1_10*C2_10)
X_10=(A1_10*B2_10)+(B1_10*D2_10)
Y_10=(C1_10*A2_10)+(D1_10*C2_10)
Z_10=(C1_10*B2_10)+(D1_10*D2_10)


S11_10 = (W_10 + (X_10/output_resistance) - (Y_10*output_resista ıce) - Z_10)/(W_10 +
(X_10/output_resistance) + (Y_10*output_resistance) + Z_10)
S12_10 = (2*((W_10*Z_10)-(X_10*Y_10)))/(W_10 + (X_10/output_resistance) +
(Y_10*output_resistance) + Z_10)
S21_10 = 2/(W_10 + (X_10/output_resistance) + (Y_10*output_resistance) + Z_10)
S22_10 = (- W_10 + (X_10/output_resistance) - (Y_10*output_resistance) + Z_10)/(W_10 +
(X_10/output_resistance) + (Y_10*output_resistance) + Z_10)


S12amp_10=sqrt((S12_10.real*S12_10.real)+(S12_10.imag*S12_10.imag))


####### end
```

```
########## plotting the graph

figure()
y=array([S12amp_0_1, S12amp_1, S12amp_2, S12amp_3, S12amp_4, S12amp_5, S12amp_6,
S12amp_7, S12amp_8, S12amp_9, S12amp_10])
t=array([0.1,1,2,3,4,5,6,7,8,9,10])
plot(t,y, 'x:')
show()
```

APPENDIX D

PLOTTING CODE FOR 3RD ORDER FILTER

```
from numpy import *
from matplotlib.pyplot import *
from math import sqrt

########## Taking input from user

pi=3.141592654
L1a=input("Enter the Inductor, L1 value: ")
C2a=input("Enter the Capacitor, C2 value: ")
L3a=input("Enter the Inductor, L3 value: ")
output_resistance=input("Enter the Output Resistance(ohm) desired: ")


######## 0.1GHZ freq

######## ABCD matrix calculation

A1_0_1 = 1
B1_0_1 = (1j*2*pi*0.1e12*L1a)
C1_0_1 = 0
D1_0_1 = 1

A2_0_1 = 1
B2_0_1 = 0
C2_0_1 = 1/(1j*2*pi*0.1e12*C2a)
D2_0_1 = 1

A3_0_1 = 1
B3_0_1 = (1j*2*pi*0.1e12*L3a)
C3_0_1 = 0
D3_0_1 = 1

######## ABCD parameter for the system cascaded

A12_0_1=(A1_0_1*A2_0_1)+(B1_0_1*C2_0_1)
B12_0_1=(A1_0_1*B2_0_1)+(B1_0_1*D2_0_1)
C12_0_1=(C1_0_1*A2_0_1)+(D1_0_1*C2_0_1)
D12_0_1=(C1_0_1*B2_0_1)+(D1_0_1*D2_0_1)

W_0_1=(A12_0_1*A3_0_1)+(B12_0_1*C3_0_1)
X_0_1=(A12_0_1*B3_0_1)+(B12_0_1*D3_0_1)
Y_0_1=(C12_0_1*A3_0_1)+(D12_0_1*C3_0_1)
Z_0_1=(C12_0_1*B3_0_1)+(D12_0_1*D3_0_1)
```

```
####### ABCD parameter is then converted to S parameter

S11_0_1 = (W_0_1 + (X_0_1/output_resistance) - (Y_0_1*output_resistance) - Z_0_1)/(W_0_1 +
(X_0_1/output_resistance) + (Y_0_1*output_resistance) + Z_0_1)
S12_0_1 = (2*((W_0_1*Z_0_1)-(X_0_1*Y_0_1)))/(W_0_1 + (X_0_1/output_resistance) +
(Y_0_1*output_resistance) + Z_0_1)
S21_0_1 = 2/(W_0_1 + (X_0_1/output_resistance) + (Y_0_1*output_resistance) + Z_0_1)
S22_0_1 = (- W_0_1 + (X_0_1/output_resistance) - (Y_0_1*output_resistance) + Z_0_1)/(W_0_1 +
(X_0_1/output_resistance) + (Y_0_1*output_resistance) + Z_0_1)


S12amp_0_1=sqrt((S12_0_1.real*S12_0_1.real)+(S12_0_1.imag*S12_0_1.imag))


####### 1GHZ freq

A1_1 = 1
B1_1 = (1j*2*pi*1e12*L1a)
C1_1 = 0
D1_1 = 1

A2_1 = 1
B2_1 = 0
C2_1 = 1/(1j*2*pi*1e12*C2a)
D2_1 = 1


A3_1 = 1
B3_1 = (1j*2*pi*1e12*L3a)
C3_1 = 0
D3_1 = 1

A12_1=(A1_1*A2_1)+(B1_1*C2_1)
B12_1=(A1_1*B2_1)+(B1_1*D2_1)
C12_1=(C1_1*A2_1)+(D1_1*C2_1)
D12_1=(C1_1*B2_1)+(D1_1*D2_1)


W_1=(A12_1*A3_1)+(B12_1*C3_1)
X_1=(A12_1*B3_1)+(B12_1*D3_1)
Y_1=(C12_1*A3_1)+(D12_1*C3_1)
Z_1=(C12_1*B3_1)+(D12_1*D3_1)



S11_1 = (W_1 + (X_1/output_resistance) - (Y_1*output_resistance) - Z_1)/(W_1 +
(X_1/output_resistance) + (Y_1*output_resistance) + Z_1)
S12_1 = (2*((W_1*Z_1)-(X_1*Y_1)))/(W_1 + (X_1/output_resistance) + (Y_1*output_resistance) +
Z_1)
S21_1 = 2/(W_1 + (X_1/output_resistance) + (Y_1*output_resistance) + Z_1)
S22_1 = (- W_1 + (X_1/output_resistance) - (Y_1*output_resistance) + Z_1)/(W_1 +
(X_1/output_resistance) + (Y_1*output_resistance) + Z_1)
```

```
S12amp_1=sqrt((S12_1.real*S12_1.real)+(S12_1.imag*S12_1.imag))


####### 2GHZ freq

A1_2 = 1
B1_2 = (1j*2*pi*2e12*L1a)
C1_2 = 0
D1_2 = 1

A2_2 = 1
B2_2 = 0
C2_2 = 1/(1j*2*pi*2e12*C2a)
D2_2 = 1


A3_2 = 1
B3_2 = (1j*2*pi*2e12*L3a)
C3_2 = 0
D3_2 = 1

A12_2=(A1_2*A2_2)+(B1_2*C2_2)
B12_2=(A1_2*B2_2)+(B1_2*D2_2)
C12_2=(C1_2*A2_2)+(D1_2*C2_2)
D12_2=(C1_2*B2_2)+(D1_2*D2_2)


W_2=(A12_2*A3_2)+(B12_2*C3_2)
X_2=(A12_2*B3_2)+(B12_2*D3_2)
Y_2=(C12_2*A3_2)+(D12_2*C3_2)
Z_2=(C12_2*B3_2)+(D12_2*D3_2)


S11_2 = (W_2 + (X_2/output_resistance) - (Y_2*output_resistance) - Z_2)/(W_2 +
(X_2/output_resistance) + (Y_2*output_resistance) + Z_2)
S12_2 = (2*((W_2*Z_2)-(X_2*Y_2)))/(W_2 + (X_2/output_resistance) + (Y_2*output_resistance) +
Z_2)
S21_2 = 2/(W_2 + (X_2/output_resistance) + (Y_2*output_resistance) + Z_2)
S22_2 = (- W_2 + (X_2/output_resistance) - (Y_2*output_resistance) + Z_2)/(W_2 +
(X_2/output_resistance) + (Y_2*output_resistance) + Z_2)


S12amp_2=sqrt((S12_2.real*S12_2.real)+(S12_2.imag*S12_2.imag))


####### 3GHZ freq

A1_3 = 1
B1_3 = (1j*2*pi*3e12*L1a)
C1_3 = 0
D1_3 = 1

A2_3 = 1
B2_3 = 0
```

```
C2_3 = 1/(1j*2*pi*3e12*C2a)
D2_3 = 1


A3_3 = 1
B3_3 = (1j*2*pi*3e12*L3a)
C3_3 = 0
D3_3 = 1

A12_3=(A1_3*A2_3)+(B1_3*C2_3)
B12_3=(A1_3*B2_3)+(B1_3*D2_3)
C12_3=(C1_3*A2_3)+(D1_3*C2_3)
D12_3=(C1_3*B2_3)+(D1_3*D2_3)


W_3=(A12_3*A3_3)+(B12_3*C3_3)
X_3=(A12_3*B3_3)+(B12_3*D3_3)
Y_3=(C12_3*A3_3)+(D12_3*C3_3)
Z_3=(C12_3*B3_3)+(D12_3*D3_3)


S11_3 = (W_3 + (X_3/output_resistance) - (Y_3*output_resistance)   Z_3)/(W_3 +
(X_3/output_resistance) + (Y_3*output_resistance) + Z_3)
S12_3 = (2*((W_3*Z_3)-(X_3*Y_3)))/(W_3 + (X_3/output_resistance) + (Y_3*output_resistance) +
Z_3)
S21_3 = 2/(W_3 + (X_3/output_resistance) + (Y_3*output_resistance) + Z_3)
S22_3 = (- W_3 + (X_3/output_resistance) - (Y_3*output_resistance) + Z_3)/(W_3 +
(X_3/output_resistance) + (Y_3*output_resistance) + Z_3)


S12amp_3=sqrt((S12_3.real*S12_3.real)+(S12_3.imag*S12_3.imag))


####### 4GHZ freq


A1_4 = 1
B1_4 = (1j*2*pi*4e12*L1a)
C1_4 = 0
D1_4 = 1

A2_4 = 1
B2_4 = 0
C2_4 = 1/(1j*2*pi*4e12*C2a)
D2_4 = 1


A3_4 = 1
B3_4 = (1j*2*pi*4e12*L3a)
C3_4 = 0
D3_4 = 1

A12_4=(A1_4*A2_4)+(B1_4*C2_4)
B12_4=(A1_4*B2_4)+(B1_4*D2_4)
```

```
C12_4=(C1_4*A2_4)+(D1_4*C2_4)
D12_4=(C1_4*B2_4)+(D1_4*D2_4)


W_4=(A12_4*A3_4)+(B12_4*C3_4)
X_4=(A12_4*B3_4)+(B12_4*D3_4)
Y_4=(C12_4*A3_4)+(D12_4*C3_4)
Z_4=(C12_4*B3_4)+(D12_4*D3_4)


S11_4 = (W_4 + (X_4/output_resistance) - (Y_4*output_resistance) - Z_4)/(W_4 +
(X_4/output_resistance) + (Y_4*output_resistance) + Z_4)
S12_4 = (2*((W_4*Z_4)-(X_4*Y_4)))/(W_4 + (X_4/output_resistance) + (Y_4*output_resistance) +
Z_4)
S21_4 = 2/(W_4 + (X_4/output_resistance) + (Y_4*output_resistance) + Z_4)
S22_4 = (- W_4 + (X_4/output_resistance) - (Y_4*output_resistance) + Z_4)/(W_4 +
(X_4/output_resistance) + (Y_4*output_resistance) + Z_4)


S12amp_4=sqrt((S12_4.real*S12_4.real)+(S12_4.imag*S12_4.imag))


####### 5GHZ freq


A1_5 = 1
B1_5 = (1j*2*pi*5e12*L1a)
C1_5 = 0
D1_5 = 1

A2_5 = 1
B2_5 = 0
C2_5 = 1/(1j*2*pi*5e12*C2a)
D2_5 = 1


A3_5 = 1
B3_5 = (1j*2*pi*5e12*L3a)
C3_5 = 0
D3_5 = 1

A12_5=(A1_5*A2_5)+(B1_5*C2_5)
B12_5=(A1_5*B2_5)+(B1_5*D2_5)
C12_5=(C1_5*A2_5)+(D1_5*C2_5)
D12_5=(C1_5*B2_5)+(D1_5*D2_5)


W_5=(A12_5*A3_5)+(B12_5*C3_5)
X_5=(A12_5*B3_5)+(B12_5*D3_5)
Y_5=(C12_5*A3_5)+(D12_5*C3_5)
Z_5=(C12_5*B3_5)+(D12_5*D3_5)
```

```
S11_5 = (W_5 + (X_5/output_resistance) - (Y_5*output_resistance) - Z_5)/(W_5 +
(X_5/output_resistance) + (Y_5*output_resistance) + Z_5)
S12_5 = (2*((W_5*Z_5)-(X_5*Y_5)))/(W_5 + (X_5/output_resistance) + (Y_5*output_resistance) +
Z_5)
S21_5 = 2/(W_5 + (X_5/output_resistance) + (Y_5*output_resistance) + Z_5)
S22_5 = (- W_5 + (X_5/output_resistance) - (Y_5*output_resistance) + Z_5)/(W_5 +
(X_5/output_resistance) + (Y_5*output_resistance) + Z_5)


S12amp_5=sqrt((S12_5.real*S12_5.real)+(S12_5.imag*S12_5.imag))

.

####### 6GHZ freq

A1_6 = 1
B1_6 = (1j*2*pi*6e12*L1a)
C1_6 = 0
D1_6 = 1

A2_6 = 1
B2_6 = 0
C2_6 = 1/(1j*2*pi*6e12*C2a)
D2_6 = 1


A3_6 = 1
B3_6 = (1j*2*pi*6e12*L3a)
C3_6 = 0
D3_6 = 1

A12_6=(A1_6*A2_6)+(B1_6*C2_6)
B12_6=(A1_6*B2_6)+(B1_6*D2_6)
C12_6=(C1_6*A2_6)+(D1_6*C2_6)
D12_6=(C1_6*B2_6)+(D1_6*D2_6)


W_6=(A12_6*A3_6)+(B12_6*C3_6)
X_6=(A12_6*B3_6)+(B12_6*D3_6)
Y_6=(C12_6*A3_6)+(D12_6*C3_6)
Z_6=(C12_6*B3_6)+(D12_6*D3_6)


S11_6 = (W_6 + (X_6/output_resistance) - (Y_6*output_resistance) - Z_6)/(W_6 +
(X_6/output_resistance) + (Y_6*output_resistance) + Z_6)
S12_6 = (2*((W_6*Z_6)-(X_6*Y_6)))/(W_6 + (X_6/output_resistance) + (Y_6*output_resistance) +
Z_6)
S21_6 = 2/(W_6 + (X_6/output_resistance) + (Y_6*output_resistance) + Z_6)
S22_6 = (- W_6 + (X_6/output_resistance) - (Y_6*output_resistance) + Z_6)/(W_6 +
(X_6/output_resistance) + (Y_6*output_resistance) + Z_6)


S12amp_6=sqrt((S12_6.real*S12_6.real)+(S12_6.imag*S12_6.imag))
```

```
####### 7GHZ freq


A1_7 = 1
B1_7 = (1j*2*pi*7e12*L1a)
C1_7 = 0
D1_7 = 1

A2_7 = 1
B2_7 = 0
C2_7 = 1/(1j*2*pi*7e12*C2a)
D2_7 = 1


A3_7 = 1
B3_7 = (1j*2*pi*7e12*L3a)
C3_7 = 0
D3_7 = 1

A12_7=(A1_7*A2_7)+(B1_7*C2_7)
B12_7=(A1_7*B2_7)+(B1_7*D2_7)
C12_7=(C1_7*A2_7)+(D1_7*C2_7)
D12_7=(C1_7*B2_7)+(D1_7*D2_7)


W_7=(A12_7*A3_7)+(B12_7*C3_7)
X_7=(A12_7*B3_7)+(B12_7*D3_7)
Y_7=(C12_7*A3_7)+(D12_7*C3_7)
Z_7=(C12_7*B3_7)+(D12_7*D3_7)


S11_7 = (W_7 + (X_7/output_resistance) - (Y_7*output_resistance) - Z_7)/(W_7 +
(X_7/output_resistance) + (Y_7*output_resistance) + Z_7)
S12_7 = (2*((W_7*Z_7)-(X_7*Y_7)))/(W_7 + (X_7/output_resistance) + (Y_7*output_resistance) +
Z_7)
S21_7 = 2/(W_7 + (X_7/output_resistance) + (Y_7*output_resistance) + Z_7)
S22_7 = (- W_7 + (X_7/output_resistance) - (Y_7*output_resistance) + Z_7)/(W_7 +
(X_7/output_resistance) + (Y_7*output_resistance) + Z_7)


S12amp_7=sqrt((S12_7.real*S12_7.real)+(S12_7.imag*S12_7.imag))


####### 8GHZ freq


A1_8 = 1
B1_8 = (1j*2*pi*8e12*L1a)
C1_8 = 0
D1_8 = 1

A2_8 = 1
B2_8 = 0
```

```
C2_8 = 1/(1j*2*pi*8e12*C2a)
D2_8 = 1


A3_8 = 1
B3_8 = (1j*2*pi*8e12*L3a)
C3_8 = 0
D3_8 = 1

A12_8=(A1_8*A2_8)+(B1_8*C2_8)
B12_8=(A1_8*B2_8)+(B1_8*D2_8)
C12_8=(C1_8*A2_8)+(D1_8*C2_8)
D12_8=(C1_8*B2_8)+(D1_8*D2_8)


W_8=(A12_8*A3_8)+(B12_8*C3_8)
X_8=(A12_8*B3_8)+(B12_8*D3_8)
Y_8=(C12_8*A3_8)+(D12_8*C3_8)
Z_8=(C12_8*B3_8)+(D12_8*D3_8)


S11_8 = (W_8 + (X_8/output_resistance) - (Y_8*output_resistance) - Z_8)/(W_8 +
(X_8/output_resistance) + (Y_8*output_resistance) + Z_8)
S12_8 = (2*((W_8*Z_8)-(X_8*Y_8)))/(W_8 + (X_8/output_resistance) + (Y_8*output_resistance) +
Z_8)
S21_8 = 2/(W_8 + (X_8/output_resistance) + (Y_8*output_resistance) + Z_8)
S22_8 = (- W_8 + (X_8/output_resistance) - (Y_8*output_resistance) + Z_8)/(W_8 +
(X_8/output_resistance) + (Y_8*output_resistance) + Z_8)


S12amp_8=sqrt((S12_8.real*S12_8.real)+(S12_8.imag*S12_8.imag))


######## 9GHZ freq



A1_9 = 1
B1_9 = (1j*2*pi*9e12*L1a)
C1_9 = 0
D1_9 = 1

A2_9 = 1
B2_9 = 0
C2_9 = 1/(1j*2*pi*9e12*C2a)
D2_9 = 1


A3_9 = 1
B3_9 = (1j*2*pi*9e12*L3a)
C3_9 = 0
D3_9 = 1

A12_9=(A1_9*A2_9)+(B1_9*C2_9)
```

```
B12_9=(A1_9*B2_9)+(B1_9*D2_9)
C12_9=(C1_9*A2_9)+(D1_9*C2_9)
D12_9=(C1_9*B2_9)+(D1_9*D2_9)


W_9=(A12_9*A3_9)+(B12_9*C3_9)
X_9=(A12_9*B3_9)+(B12_9*D3_9)
Y_9=(C12_9*A3_9)+(D12_9*C3_9)
Z_9=(C12_9*B3_9)+(D12_9*D3_9)


S11_9 = (W_9 + (X_9/output_resistance) - (Y_9*output_resistance) - Z_9)/(W_9 +
(X_9/output_resistance) + (Y_9*output_resistance) + Z_9)
S12_9 = (2*((W_9*Z_9)-(X_9*Y_9)))/(W_9 + (X_9/output_resistance) + (Y_9*output_resistance) +
Z_9)
S21_9 = 2/(W_9 + (X_9/output_resistance) + (Y_9*output_resistance) + Z_9)
S22_9 = (- W_9 + (X_9/output_resistance) - (Y_9*output_resistance) + Z_9)/(W_9 +
(X_9/output_resistance) + (Y_9*output_resistance) + Z_9)


S12amp_9=sqrt((S12_9.real*S12_9.real)+(S12_9.imag*S12_9.imag))


####### 10GHZ freq


A1_10 = 1
B1_10 = (1j*2*pi*10e12*L1a)
C1_10 = 0
D1_10 = 1

A2_10 = 1
B2_10 = 0
C2_10 = 1/(1j*2*pi*10e12*C2a)
D2_10 = 1


A3_10 = 1
B3_10 = (1j*2*pi*10e12*L3a)
C3_10 = 0
D3_10 = 1

A12_10=(A1_10*A2_10)+(B1_10*C2_10)
B12_10=(A1_10*B2_10)+(B1_10*D2_10)
C12_10=(C1_10*A2_10)+(D1_10*C2_10)
D12_10=(C1_10*B2_10)+(D1_10*D2_10)


W_10=(A12_10*A3_10)+(B12_10*C3_10)
X_10=(A12_10*B3_10)+(B12_10*D3_10)
Y_10=(C12_10*A3_10)+(D12_10*C3_10)
Z_10=(C12_10*B3_10)+(D12_10*D3_10)
```

```
S11_10 = (W_10 + (X_10/output_resistance) - (Y_10*output_resistance) - Z_10)/(W_10 +
(X_10/output_resistance) + (Y_10*output_resistance) + Z_10)
S12_10 = (2*((W_10*Z_10)-(X_10*Y_10)))/(W_10 + (X_10/output_resistance) +
(Y_10*output_resistance) + Z_10)
S21_10 = 2/(W_10 + (X_10/output_resistance) + (Y_10*output_resistance) + Z_10)
S22_10 = (- W_10 + (X_10/output_resistance) - (Y_10*output_resistance) + Z_10)/(W_10 +
(X_10/output_resistance) + (Y_10*output_resistance) + Z_10)


S12amp_10=sqrt((S12_10.real*S12_10.real)+(S12_10.imag*S12_10.imag))


####### end

########## plotting the graph

figure()
y=array([S12amp_0_1, S12amp_1, S12amp_2, S12amp_3, S12amp_4, S12amp_5, S12amp_6,
S12amp_7, S12amp_8, S12amp_9, S12amp_10])
t=array([0.1,1,2,3,4,5,6,7,8,9,10])
plot(t,y, 'x:')
show()
```